



UNIVERSITÉ DE  
MONTPELLIER

# École Polytechnique Universitaire de Montpellier

## *Description de projet*

### *Projet S5*

***DEBRYDE – Suivi de fermentation***

***Réalisé le 02/10/23***



**MEA**



### **Groupe G4 :**

- Pierre BOUCKSON
- Théo FOULQUIER
- Khaoula LAHBIB
- Mathieu RICHARD
- Edgars VASILJEVS
- Maxime VITAL
- Ambre VUAILLAT
- Bilel ZAKANI-FADILI

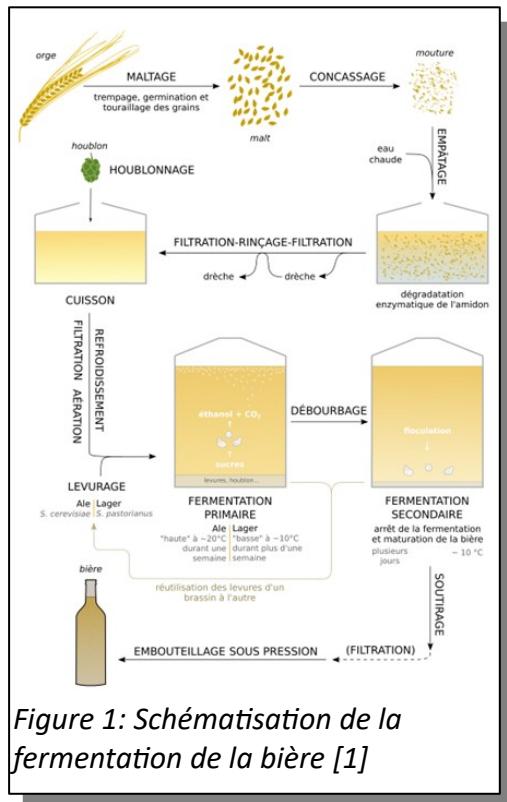
### **Rapport Intermédiaire du projet**

Département Électronique et Informatique Industrielle,  
Polytech Montpellier, Université de Montpellier

## *Tables des matières*

- *Introduction*
- *I – Gestion du projet*
  - *I.1 – Présentation de l'équipe*
  - *I.2 – Cahier des charges*
  - *I.3 – Création du diagramme*
  - *I.4 – Répartition des tâches*
  - *I.5 – Gestion du temps*
- *II – Architecture*
- *III – Turbidité*
  - *III.1 – Introduction*
  - *III.2 – Solutions proposées*
  - *III.3 – Code ARDUINO*
  - *III.4 – Tests*
  - *III.5 – Conclusion*
- *IV – Température*
  - *IV.1 – Introduction*
  - *IV.2 – Solution retenue*
  - *IV.3 – Code ARDUINO + Tests + Difficulté*
  - *IV.4 – Conclusion + Analyse*
- *V – pH*
  - *V.1 – Introduction*
  - *V.2 – Fonctionnement*
  - *V.3 – Conclusion*
- *VI – Densité + Flottabilité*
  - *VI.1 – Introduction*
  - *VI.2 – Fonctionnement + Analyse*
  - *VI.3 – Test + Conclusion*
- *VII – Batterie + Antenne Relais*
  - *VII.1 – Introduction*
  - *VII.2 – Fonctionnement + Analyse*
  - *VII.3 – Test + Conclusion*
- *Mode d'emploi*
- *Retour d'expérience*
- *Conclusion*

## ➤ Introduction



Lors de la conception de la bière, la phase la plus importante est celle de la fermentation. Lors de la fermentation [1], il ne faut pas négliger certains aspects qui doivent être respectés pour pouvoir avoir une bonne bière, sachant qu'il faut respecter la durée de la conception de la bière.

Le principe de la fermentation de la bière est expliqué sur la Figure 1, mais la partie qui pose des problèmes lors de cette conception est les deux phases de fermentation de la bière avec :

Fermentation primaire : les différents ingrédients seront mélangés à une température ordonnée par le fabricant

Fermentation secondaire : après le mélange, la bière se reposera pendant quelque temps à une température donnée par le fabricant

Lors de ce projet pour notre 3<sup>ème</sup> année d'école d'ingénieur dans la filière MEA (Microélectronique et Automatique), nous allons chercher des solutions permettant de prévenir si les conditions de la fermentation sont respectées en nous aidant des capteurs qui mesureront les aspects que nous voudrons mesurer.

Ce projet a eu pour but de nous familiariser à la fermentation de la bière et de nous faire comprendre le rôle d'un ingénieur.

Pour pouvoir aider le fabricant à obtenir une bonne bière, nous avions dû réfléchir aux différentes phases de la fermentation qui doivent être respecté et dont nous pouvons faire des mesures pouvant être utile au fabricant.

## ➤ I – Gestion du projet

### o I.1 – Présentation de l'équipe

#### 1. Pierre BOUCKSON



Avant d'intégrer Polytech Montpellier, j'ai réalisé une PEIP type physique à Lille où j'ai acquis des compétences variées aussi bien en électronique qu'en physique et mécanique. J'ai décidé de rejoindre la spécialité Microélectronique et Automatique de Montpellier, car depuis tout temps, je suis passionné d'électronique ayant même réalisé quelques projets personnels à l'aide d'une Arduino et du C++.

J'essaye donc au mieux au sein de ce projet de mettre à profit mes compétences afin d'aider au mieux à la réussite de ce projet.

#### 2. Théo FOULQUIER



J'ai décidé d'intégrer la filière MEA (Microélectronique et Automatique) suite à deux années en PEIP\_A à Montpellier. Ce cursus m'a permis de renforcer mes connaissances en physique, mathématiques, chimie, et informatique. J'ai ainsi eu une formation adaptée à ma filière.

Durant mon parcours scolaire, j'ai participé à plusieurs projets de groupe exigeant notamment des compétences en mécanique, électronique et informatique. J'ai également appris les bases du langage C++ en autodidacte. De plus, mes expériences professionnelles et associatives me permettent de facilement m'intégrer dans un groupe et d'y prendre les responsabilités si besoin.

#### 3. Khaoula LAHBIB

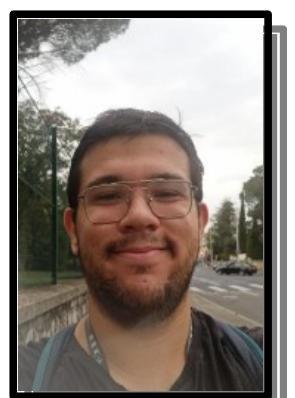


Avant d'intégrer la filière Microélectronique et Automatique à Polytech Montpellier, j'ai effectué un DUT GÉNIE Électronique et Informatique Industrielle qui m'a permis d'acquérir de nombreuses compétences théoriques et techniques en électronique et en langage C.

J'ai effectué deux stages pendant mon DUT qui m'ont permis de travailler en équipe ainsi que de développer des techniques de travail.

La dernière année en MEA3 , en effectuant deux projets semestriels : 'L'oscilloscope cathodique' et 'Indicateur de liquide' , j'ai pu acquérir des compétences en électronique analogique , dans la manipulation des capteurs et de différentes cartes électroniques: 'ESP32' , 'Arduino' .

#### 4. Mathieu RICHARD



Avant d'intégrer l'école de Polytech Montpellier en spécialité Microélectronique et Automatique, j'ai passé trois ans en classe préparatoire scientifique à Valence. J'y ai acquis de solides connaissances en physique et en mathématiques, ainsi qu'une bonne maîtrise du langage Python. Enfin, j'ai appris grâce à un projet se déroulant toute l'année se nommant TIPE à m'organiser.

Durant ce projet, qui est ma première expérience de travail en groupe, je souhaite développer mes compétences en matière de communication et de gestion de projet.

Compétences apportées : je compte apporter ma compréhension de l'environnement Arduino ainsi que ma rigueur et mon sens de l'organisation au sein de l'équipe.

## **5. Edgars VASILJEVS**



Passionné par les sciences en général, j'ai décidé de suivre un parcours scientifique dans mes études. Parmi tous les métiers possibles, j'ai décidé de m'orienter vers celui d'ingénieur, spécialisé en Microélectronique et Automatique en intégrant le parcours de Polytech Montpellier dans le département MEA, car je voulais y développer des compétences en informatique et en électronique. Avant cela, j'ai effectué deux ans en école préparatoire intégrée en PEIP\_A à Montpellier durant lesquelles j'ai pu renforcer mes compétences en mathématiques et physique générales en plus de l'informatique.

J'ai déjà réalisé des travaux de groupes centrés essentiellement sur le travail de recherches d'informations pour développer un sujet. Autrement, il s'agit de mon tout premier projet en lien avec l'électronique et j'espère y acquérir des compétences qui me seront utiles plus tard.

## **6. Maxime VITAL**



Avant d'intégrer l'école de Polytech Montpellier en spécialité Microélectronique et Automatique, j'ai passé deux ans en classe préparatoire scientifique et technologique à Toulouse. J'y ai acquis de solides connaissances en physique, en mathématiques, en électronique et en mécanique ainsi qu'une bonne maîtrise du langage Python. Enfin, j'ai appris à m'organiser dans des projets lors de mon TIPE en prépa et de mon projet de terminale en STI2D, et c'est en STI2D que j'ai appris à utiliser le diagramme de Gantt.

Compétences apportées : je sais organiser un groupe sur les délais à respecter, et sur le domaine des codes Arduino.

## **7. Ambre VUAILLAT**



Après l'obtention de mon bac, j'ai suivi la formation PEIP à Polytech Nice Sophia ce qui m'a permis ensuite d'intégrer Polytech Montpellier au sein de la filière Microélectronique et Automatique.

La formation PEIP m'a permis d'acquérir des bases en mathématiques, en physique, en informatique (Python, Java, HTML) et en électronique. J'ai eu l'opportunité de réaliser des projets en informatique et un projet en électronique durant lequel j'ai réalisé un bras robotisé. Ces projets m'ont permis de développer mes compétences de travail en groupe et d'organisation, et d'acquérir de nombreuses connaissances.

## **8. Bilel ZAKANI-FADILI**



Ayant toujours eu une passion pour les métiers de l'électronique et de l'informatique, je voulais imaginer, concevoir mais surtout innover. Pour réaliser à terme ce projet, je dois acquérir une connaissance pointue dans les matières scientifiques ainsi qu'une rigueur irréprochable, et cela, dès la seconde.

Après obtention d'un bac général avec spécialités : Maths, Physique-Chimie et NSI. J'ai décidé de poursuivre mon cursus à Polytech en commençant par la PEIP à Marseille et par la suite, la filière Microélectronique et Automatique à Montpellier. J'ai déjà eu la chance de travailler sur un projet (baby-foot connecté) comme celui-ci utilisant comme base une carte Arduino.

Ce projet est tout fois différent car l'on se rapproche grandement d'un travail attendu d'un ingénieur et cela demande donc de la rigueur, de la communication, de l'autonomie et un esprit critique.

## *o 1.2 – Cahier des charges*

L'objectif principal de notre projet est de concevoir et de réaliser un système équipé de capteurs permettant de suivre l'évolution de la fermentation de la bière dans une micro-brasserie en mesurant des caractéristiques.

Ces capteurs doivent nous permettre de suivre lors de la fermentation le taux de sucre, la température et la clarté de la bière. Puis ils doivent nous donner les mesures en temps réel même si la boîte est immergée dans la cuve.

Nous constatons que notre cahier des charges doit respecter les différentes caractéristiques techniques requises qui sont les suivants :

### **1. Capteur de température :**

Notre dispositif devra être capable de mesurer la température ambiante à l'intérieur de la cuve de manière précise. La plage de température que nous souhaitons mesurer se situe entre 18 et 24°C. On devra donc avoir un capteur avec une incertitude pas très élevée pour éviter des marges d'erreur trop grande par rapport à notre plage de température. De plus, le capteur devra être capable de communiquer la température relevée lors de la fermentation à travers l'antenne.

### **2. Capteur de turbidité :**

Notre dispositif devra être capable de quantifier les particules présentes dans notre solution, offrant ainsi à l'utilisateur la possibilité de suivre de près le processus de fermentation et il devra aussi être capable de préciser une plage de mesure minimale et maximale avec une incertitude relative. Afin de visualiser de manière adéquate l'évolution de sa solution, nous devrons faire des analyses en terme temporel, car nous allons pouvoir observer une variation du nombre de particules en fonction du temps. Pour pouvoir faire des analyses, nous allons utiliser un capteur qui devra être capable de communiquer une mesure lisible et compréhensible à l'utilisateur.

### **3. Capteur de densité :**

Le dispositif devra être capable de mesurer de manière indirecte à travers des corrélations ou des formules qui lie la densité à une autre caractéristique que l'on peut mesurer avec un capteur. Une mesure précise ne pourra pas être obtenue, mais on prendra en compte une certaine incertitude relative dont l'utilisateur devra être informé. La plage de mesure se situe entre 1000 et 1100g/cm<sup>3</sup>. De plus, plus la valeur est élevée et plus le moût est chargé en sucre et dense. Il sera important de trouver une solution dont l'incertitude relative n'est pas très élevé pour rester dans la plage de mesure que l'on souhaite. Le capteur devra être capable de renvoyer une information à travers l'antenne qui sera comprise par l'utilisateur.

### **4. Système d'alimentation :**

Notre système devra être suffisamment alimenté par intermittence pour durer le plus longtemps possible, car la fermentation dure plusieurs semaines. Le système d'alimentation devra également pouvoir être rechargé. Une batterie avec une autonomie importante sera à privilégier pour notre objet final.

### **5. Tupperware :**

Notre système sera dans un Tupperware qui devra être entièrement étanche pour éviter d'abîmer les câblages électriques. De plus, il devra être résistant aux conditions environnementales de la fermentation de la bière. Nous sommes limités à un volume de 1 litre (hauteur = ?, largeur = ?, longueur = ?) et donc notre système devra occuper cet espace de manière optimisé.

## **6. Communication par l'antenne :**

Une antenne réalisée par le professeur nous sera fournie au cours du projet. Il sera donc important de penser à son intégration dans le Tupperware et de comprendre son fonctionnement pour pouvoir transmettre des informations, cependant, nous devrons comprendre comment les informations lui sont fournies tour à tour et comment il peut nous les transmettre. Or, il transmet nos valeurs mesurées par nos différents capteurs dans la cuve. Il faudra également penser à informer l'utilisateur sur la manière de recevoir les données des capteurs et comment il pourra les visualiser.

## **7. Documentation :**

Une fois notre système conçu et fonctionnel, il sera important de rédiger un manuel d'utilisation suffisamment détaillé pour les utilisateurs afin qu'ils comprennent l'utilité de l'objet et de comprendre comment s'en servir, mais s'en oublier de montrer étape par étape comment obtenir les valeurs mesurées sur un ordinateur par exemple. Il sera également utile de fournir un schéma électrique du système pour que les personnes expérimentées puissent réparer l'objet en cas de panne et de favoriser sa durabilité. Enfin, une liste de composants utilisés par le système sera également très utile si jamais il faut les remplacer en cas de panne.

## **8. Contraintes du projet :**

Afin de concevoir notre projet, notre objet devra respecter des normes de sécurité en vigueur dans les micro-brasseries notamment en terme alimentaire et hygiène. De plus, il devra respecter un certain budget global pour ne pas revenir très cher à l'utilisateur qui sera intéressé par notre projet.

## o I.3 – Création du diagramme

Lors de la création du diagramme de Gantt, nous avons réfléchi sur les différentes parties réalisables répondant à la problématique. Lors de la réalisation du projet, nous avons dû réaliser une répartition du travail en groupe. Pour chaque groupe, nous avons dû trouver un moyen qui nous permet de communiquer entre chaque groupe. Cette communication nous a permis de comprendre quelles sont les difficultés de chacun durant ce projet. On observe en Figure I.3.1, la réalisation du diagramme de Gantt entre la rentrée et les vacances de la Toussaint.

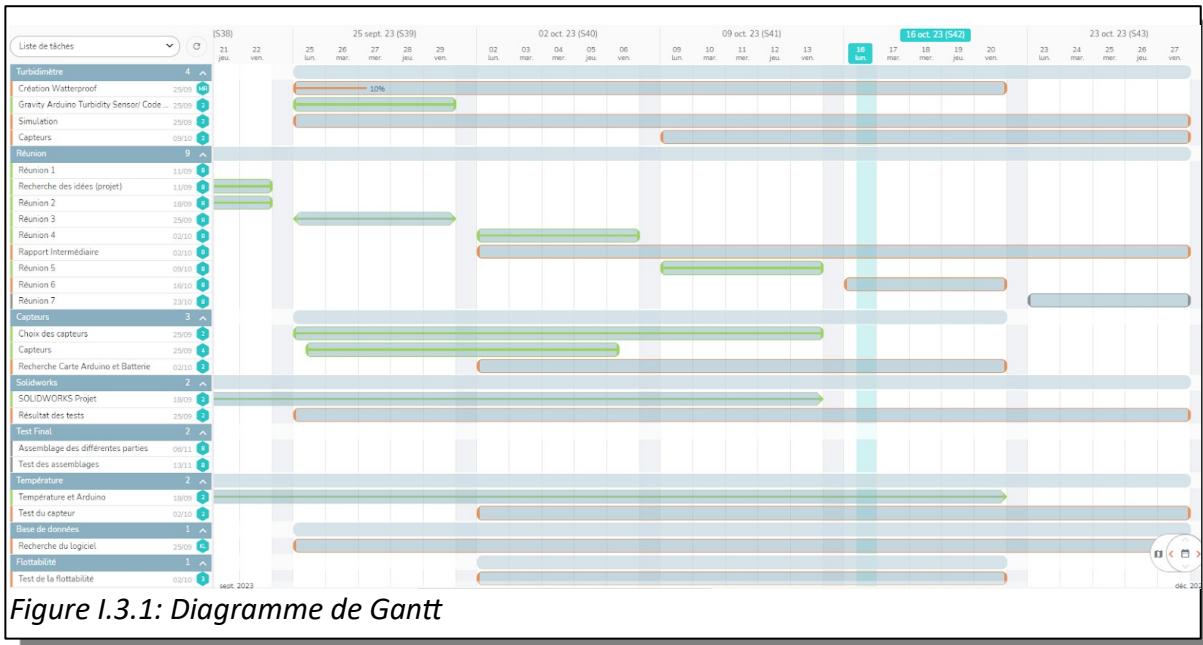


Figure I.3.1: Diagramme de Gantt

Nous avons dû approfondir le diagramme de Gantt, car nous avons réalisé, pendant la période entre la rentrée et les vacances de la Toussaint, que le projet avançait super, vite et qu'il nous fallait approfondir le travail pour pouvoir avoir un produit fini en fin de projet le 17/01/24. Cette organisation s'est approfondie et nous pouvons l'observer dans la partie Annexe à la fin du rapport sur les Figure 2 et 3. Lors de la progression de notre projet, nous avons dû rajouter des tâches supplémentaires qui nous permettent d'anticiper sur des erreurs éventuelles.

## o I.4 – Répartition des tâches

On a réparti le groupe en fonction des tâches attribuées à chacun et chacune :

- Premier Groupe : Théo FOULQUIER et Ambre VUAILLAT
- Deuxième Groupe : Mathieu RICHARD et Edgars VASILJEVS
- Troisième Groupe : Pierre BOUCKSON et Khaoula LAHBIB
- Quatrième Groupe : Bilel ZAKANI-FADILI et Maxime VITAL

Lors de la réalisation des tâches, nous avons discuté sur les tâches que chacun voulait réaliser. Pour les quatre groupes, l'objectif principal était le même et c'était de répondre au problème du constructeur. Cependant, la mesure du pH était devenue compliquée, car nous avons eu quelques difficultés sur le choix du capteur par son coût et sa résistivité.

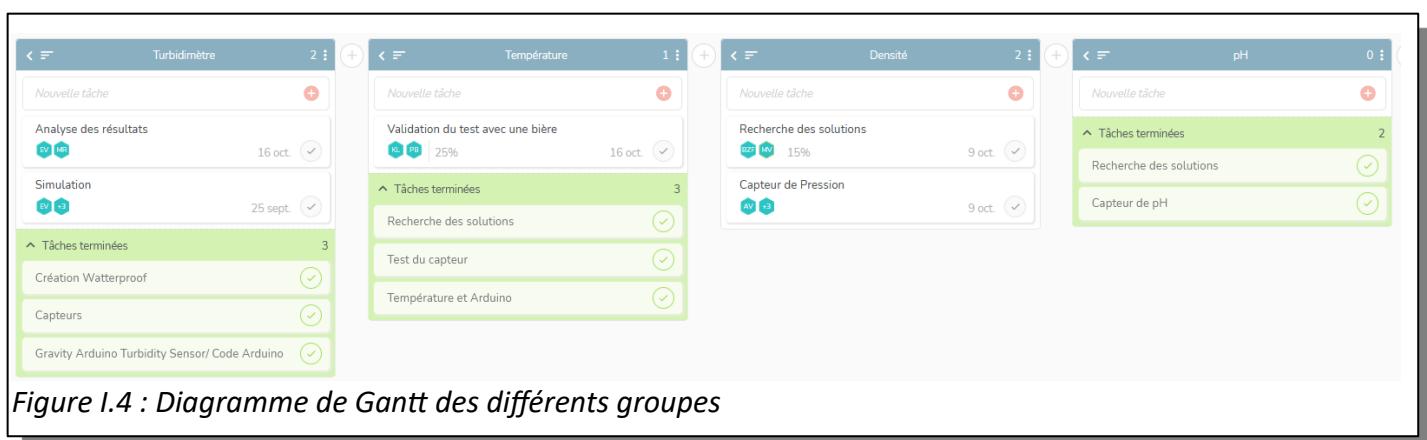
Le premier groupe est chargé de la mesure du pH. Durant ce projet, ils ont dû chercher différentes méthodes pour réaliser des mesures de pH. Après leur différente recherche, ils ont trouvé quelques solutions leur permettant de réaliser les mesures de pH. Ces recherches leur ont permis de comprendre qu'il n'y a pas qu'une seule solution leur permettant de faire des mesures de pH.

Le deuxième groupe s'occupait de la mesure de la turbidité. Pendant ce projet, ils ont dû comprendre le principe de turbidité puis de chercher les différentes solutions permettant de répondre au problème posé. Cependant, leurs recherches ont permis de trouver des solutions intéressantes.

Le troisième groupe travaillait sur la mesure de la température. Au cours du projet, ce groupe était chargé de trouver une méthode pour faire des mesures de température et de comparer la température à l'intérieur de la cuve avec celle de l'extérieur. Lors des recherches, le groupe n'a rencontré aucun problème et a réussi à programmer le capteur.

Le quatrième groupe œuvrait sur mesure de la densité. Au cours du projet, ce groupe a dû chercher une méthode permettant de faire des mesures de densité sans que la boîte ne coule au fond de la cuve. La méthode qui a été trouvée pendant ces recherches était de faire flotter la boîte pour négliger l'inclinaison de la boîte nous permettant d'obtenir une relation entre la masse volumique et l'angle de rotation de la boîte.

Nous retrouvons cette organisation des groupes sur la Figure I.4, dont on peut lire de gauche à droite la turbidité, la température, la densité et le pH.



## o 1.5 – Gestion du temps

Pour la gestion du travail, nous avons regroupé les tâches ensemble pour ne pas en laisser une à l'écart. Nous avons communiqué entre chaque partie, pour qu'on puisse être en accord sur la problématique du projet. Cette communication entre chaque partie du projet a été réalisée par des rapports hebdomadaires qui reflètent tout ce qui a été vu dans chaque séance de projet.

Pour que cette gestion du temps soit efficace, nous avons utilisé un diagramme de Gantt qui a permis de préciser à chaque membre du groupe ce qui doit être fait durant chaque séance de projet puis de savoir quelles sont les tâches qu'ils doivent être réalisées.

Nous avons constaté que pour éviter davantage erreurs, il a fallu mettre à jour à chaque séance le diagramme de Gantt et à chaque réunion de faire le point sur ce qui a été fait et ce qui doit être réalisé pour la prochaine séance.

Cette organisation nous a permis d'éviter de perdre du temps sur les heures de travail de chaque groupe avant d'arriver à la date limite du projet qui est le 19/01/2024.

Maxime a su prendre en main à la perfection l'outil permettant la réalisation du diagramme de Gantt (Beesbusy), il a ainsi assuré la bonne gestion et coordination des différentes parties pour que l'avancée du groupe n'en soit pas affectée.

Nous avons observé que dans les diagrammes de Gantt présentés dans l'annexe sous la forme de Figure 2 et 3, que l'ensemble du groupe était divisé en petites équipes afin de faire progresser le projet jusqu'à son terme, qui consistait à fournir des informations au fabricant concernant la température, le taux de sucre et la clarté de la bière pendant la fermentation.

Cependant, cette organisation des tâches a eu quelques complications au démarrage. Nous avons rencontré des problèmes lors de la recherche des différents principes pour la conception d'une bière. Cependant, les différents éléments majeurs à sa conception nous ont posé quelques problèmes, car il nous fallait trouver les différents éléments nous permettent de répondre aux exigences du constructeur. Suite à de nombreuses recherches, nous avons pu trouver les différents éléments qui sont :

- ➔ la température
- ➔ la turbidité
- ➔ la densité
- ➔ le pH

Or, nous avons constaté que pour faire des mesures de densité, nous avions besoin que la boîte soit immergée dans la cuve remplie de bière, sans que celle-ci ne coule pas jusqu'au fond. Nous avons dû faire des recherches sur la flottabilité de la boîte dans la cuve et mais aussi une recherche sur une batterie permettant au système de fonctionner pendant quelques jours ou même bien quelques semaines.

## ➤ II – Architecture

Notre prototype va prendre la forme d'une boîte dont on peut observer beaucoup mieux son aspect dans la Figure II.1.

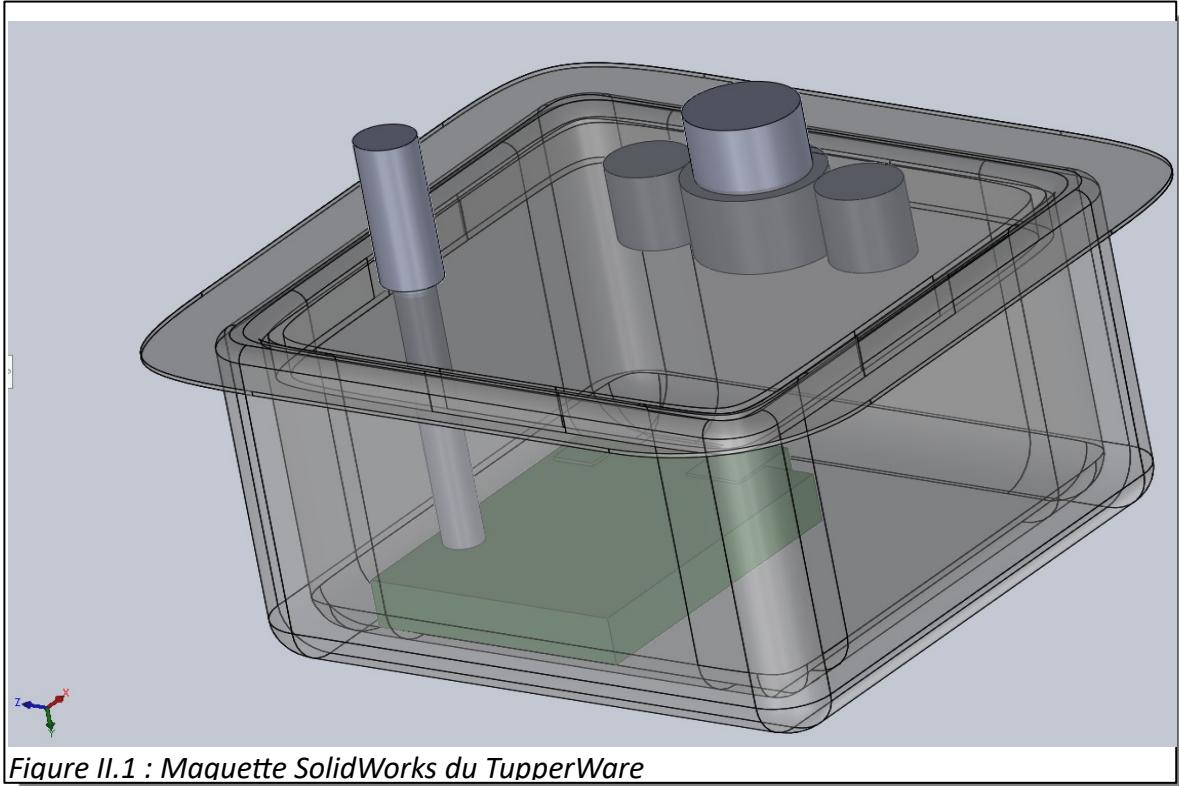


Figure II.1 : Maquette SolidWorks du TupperWare

Cette boîte a été modélisée sur SolidWorks (Figure II.1), cette modélisation a permis de comprendre comment nous allons pouvoir réaliser des mesures dans la cuve, mais aussi quelles mesures nous devrions faire pour concevoir une bière d'exception.

Nous avons remarqué que les différentes mesures (Figure II.2) que nous allons réaliser sont :

- Capteur de température
- Capteur de turbidité
- Capteur de densité

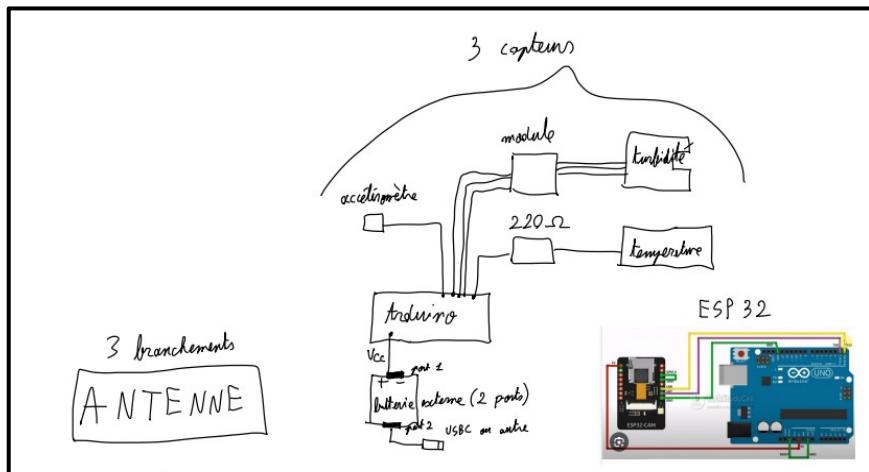


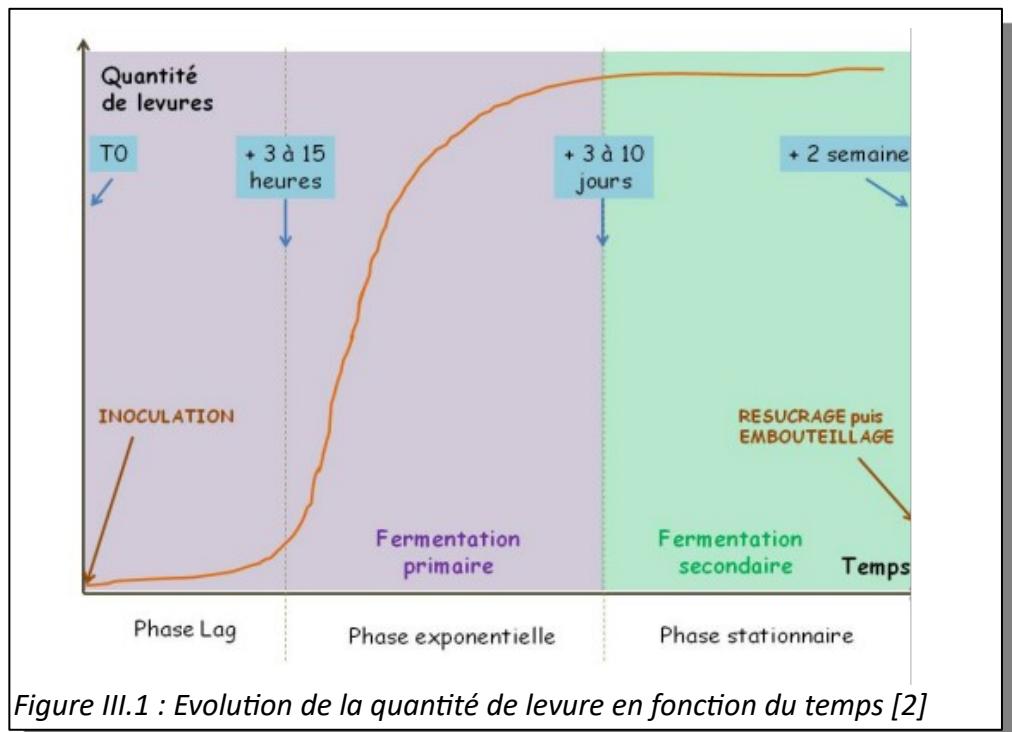
Figure II.2 : Schéma électrique des différentes capteurs

## ➤ III – Turbidité

### ○ III.1 – Introduction

Le but du capteur de turbidité est de pouvoir capter l'évolution du nombre de particules dans la solution. La mesure de ce nombre de particules permet de tracer une courbe permettant au brasseur de vérifier son moût. Pour mieux comprendre le choix de ce capteur, il faut savoir qu'au cours de la fermentation de la bière, le nombre de particules en suspension évolue. Au début, il y a peu de particules. Ensuite, la levure se multiplie de façon exponentielle, consommant les sucres et produisant de l'alcool, avant de ralentir à mesure que la fermentation progresse.

Normalement, le capteur devra relever une courbe de ce type (Figure III.1.1) :



### ○ III.2 – Solutions proposées

Nous avons d'abord envisagé d'utiliser des capteurs infrarouges [3], avec un émetteur et un récepteur placés en face l'un de l'autre, séparés par la solution. L'idée était de pouvoir vérifier que plus le nombre de particules augmentait, plus la solution devenait opaque, réduisant ainsi la quantité d'information captée par le récepteur [2]. Malheureusement, cette approche s'est avérée assez peu efficace au démarrage, car nous avons constaté quelques erreurs de manipulation du capteur lors des nombreux tests réalisés.

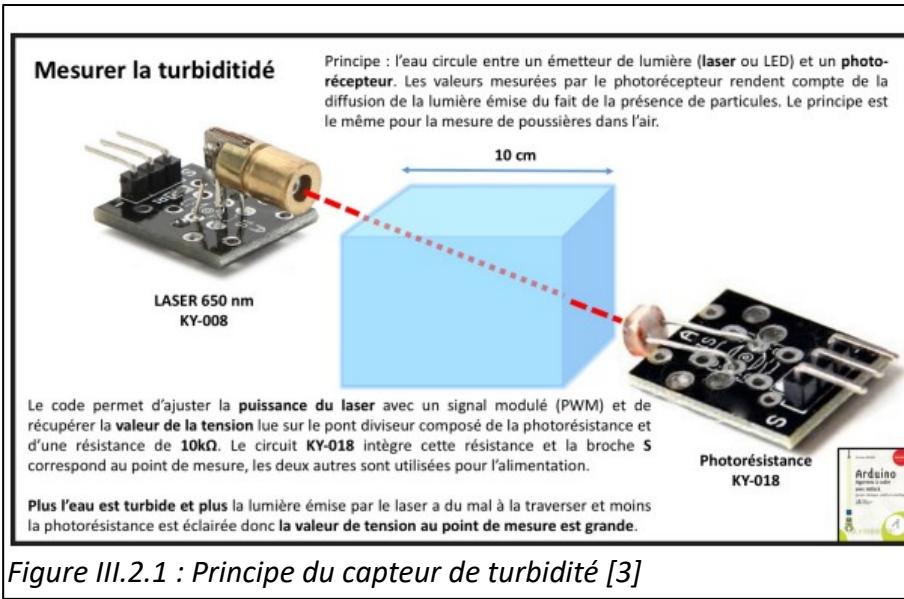


Figure III.2.1 : Principe du capteur de turbidité [3]

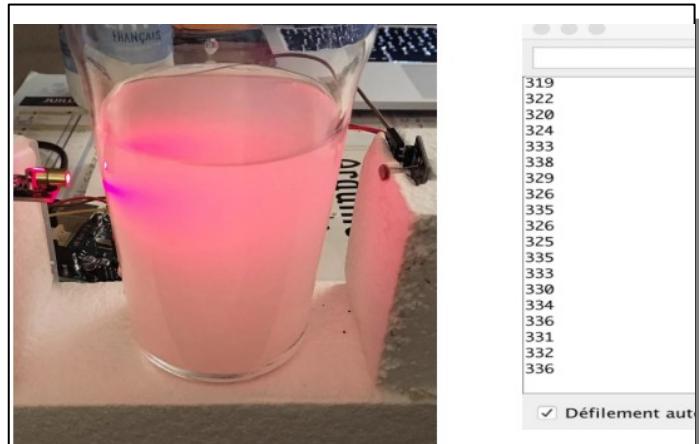


Figure III.2.2 : Test du capteur [4]

La solution adaptée pour faire une analyse de la turbidité lors de la fermentation est un capteur de turbidité [3]. Un capteur de turbidité mesure la quantité de particules en suspension dans un liquide en utilisant une source lumineuse (généralement une LED) pour émettre de la lumière à travers le liquide. Un récepteur de lumière, tel qu'une photodiode, mesure l'intensité lumineuse qui atteint son capteur. Lorsque des particules en suspension dans le liquide dispersent la lumière, l'intensité lumineuse diminue. Cette mesure est convertie en une valeur de turbidité à l'aide d'un étalonnage préalable. Les capteurs de turbidité sont essentiels pour surveiller la qualité de l'eau dans diverses applications, en fournissant une indication de la clarté du liquide. Enfin, en comparaison à la solution précédente, le capteur est plus précis, car nous avons constaté qu'il nous donnait des informations en binaire, mais plus précisément des valeurs allant entre 0 et 1024, mais on peut l'observer sur la Figure.III.2.2 que plus la valeur se rapproche de 1024 et plus le liquide est clair et peut transmettre une information sans qu'il ne puisse en perdre en cours de route.

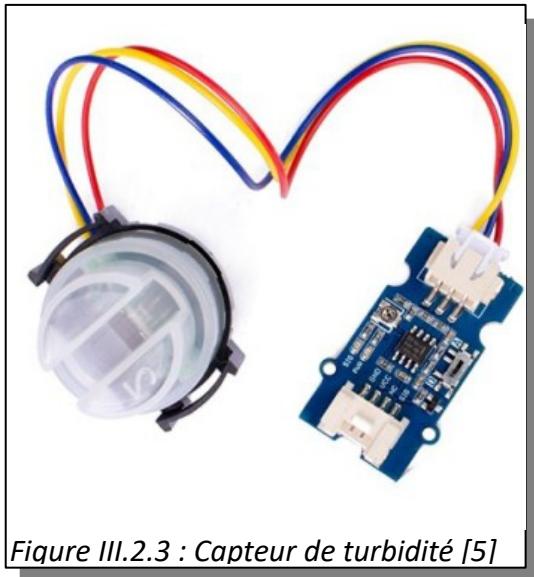


Figure III.2.3 : Capteur de turbidité [5]

### o III.3 – Code ARDUINO

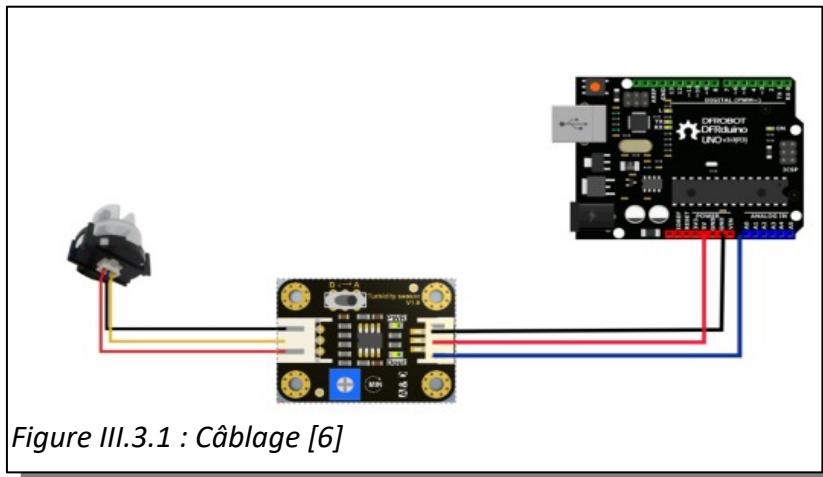


Figure III.3.1 : Câblage [6]

```

void setup() {
    Serial.begin(9600); //Baud rate: 9600
}
void loop() {
    int sensorValue = analogRead(A0); // read the input on analog pin 0:
    float voltage = sensorValue * (5.0 / 1024.0); // Convert the analog reading
    // which goes from 0 - 1023 to a voltage (0 - 5V):
    Serial.println(voltage); // print out the value you read:
    delay(500);
}

```

Figure III.3.2 : Code Arduino

### o III.4 – Tests

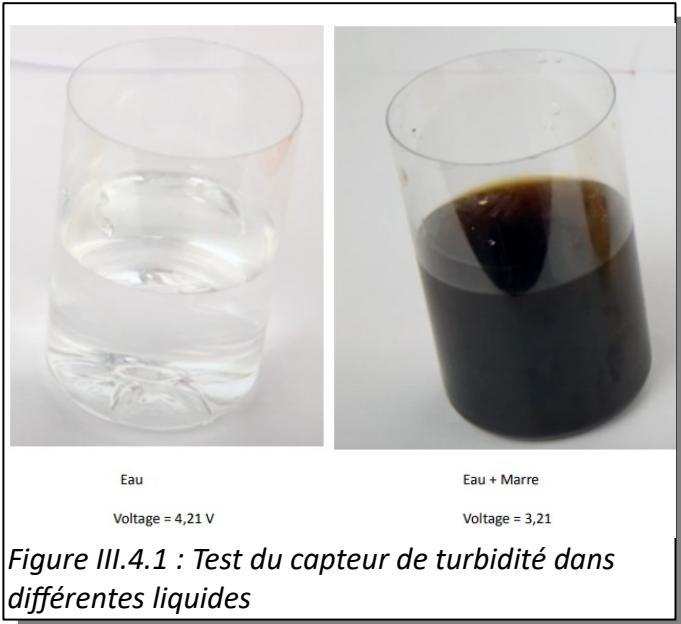


Figure III.4.1 : Test du capteur de turbidité dans différentes liquides

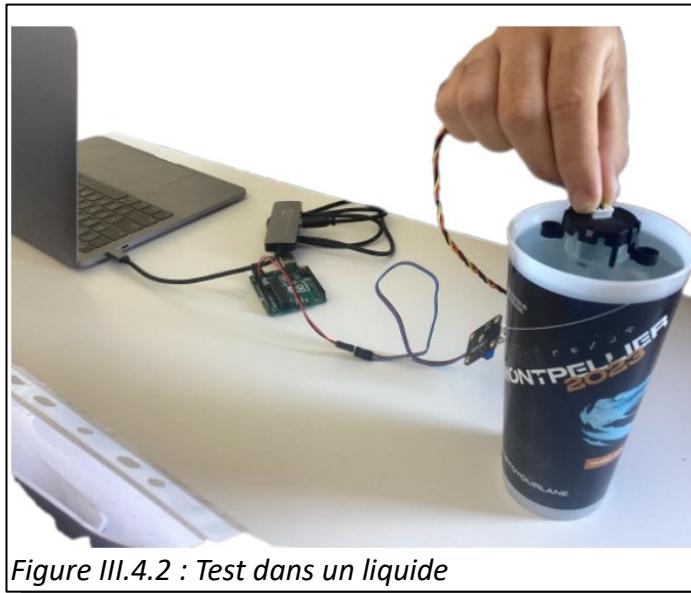


Figure III.4.2 : Test dans un liquide

Le test (Figure III.4.2) a été couronné de succès, mais il est important de noter qu'une variation de luminosité peut avoir un impact significatif sur les valeurs de tension, non seulement plus il y a de particules dans la solution et plus la luminosité qui traversera la solution sera faible. Pour résoudre ce problème qui n'était pas pris en compte dans la fiche technique, nous avons prévu d'ajouter une LED qui sera utile pour pouvoir nous informer si la lumière traverse la solution sans problème, et cela nous a permis aussi de maintenir une luminosité constante dans n'importe quel environnement.

### o III.5 – Conclusion

Le capteur répond parfaitement à nos exigences en termes de performances et de précision. Il n'est pas nécessaire d'ajouter une LED pour assurer une lumière constante. La dernière étape cruciale pour ce capteur consiste à l'intégrer dans le boîtier et de le rendre étanche à l'eau.

## ➤ IV – Température

### o IV.1 – Introduction

La température est un élément central de la fermentation, elle doit être régulée le plus possible pour éviter toute perturbation extérieure pouvant entraver le déroulement de la fermentation.

C'est donc grâce à la mesure du capteur de température que la régulation peut s'effectuer.

### o IV.2 – Solution retenue

Nous avons utilisé le capteur DS18B20 acheté sur Farnell :

Il possède 3 Pin :

- +3V, 5,5 V
- Signal
- GND

#### SPECIFICATION

- Usable with 3.0V to 5.5V power/data
- ±0.5°C Accuracy from -10°C to +85°C
- Usable temperature range: -55 to 125°C (-67°F to +257°F)
- 9 to 12 bit selectable resolution
- Uses 1-Wire interface- requires only one digital pin for communication
- Unique 64 bit ID burned into the chip
- Multiple sensors can share one pin
- Temperature-limit alarm system
- Query time is less than 750ms
- 3 wires interface:

Figure IV.2 : Fiche technique du capteur DS18B20 [7]

La fiche technique du fabricant (Figure IV.2) nous a informées sur l'utilisation du capteur. Il nous a permis de comprendre comment le capteur peut être fonctionnel sur une plage de température entre -55° et 125°, pas de souci donc à ce niveau pour notre fermentation.

L'erreur à ne pas faire est de ne pas mettre de résistance entre le Pin 5V et le Pin Signal, cela risque d'endommager le capteur, ce capteur résiste bien à l'eau et donc il peut être utilisé dans la bière. Sa longue taille permet de le faire tremper dans la bière en toute circonstance que l'on choisisse à l'avenir de fixer à la boîte ou de le laisser flotter ou même de faire varier sa hauteur, le capteur sera toujours plongé dans la bière.

### o IV.3 – Code Arduino + Tests + Difficultés

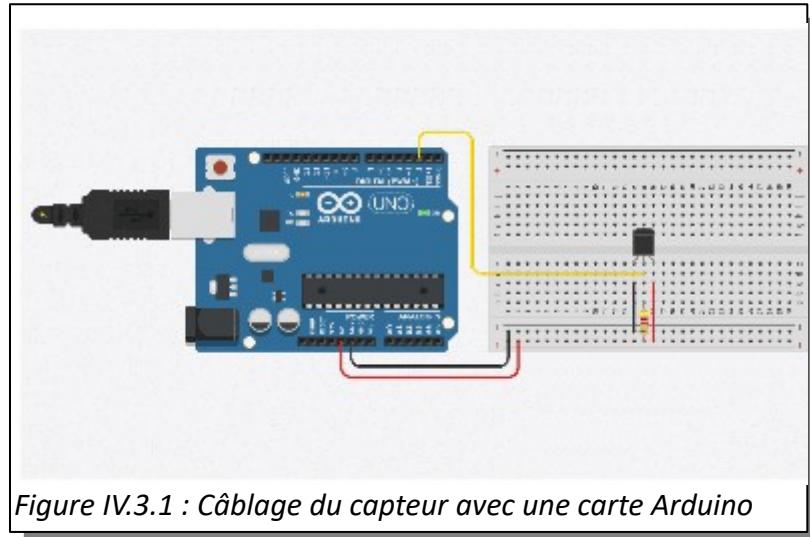


Figure IV.3.1 : Câblage du capteur avec une carte Arduino

Ici (Figure IV.3.1), le capteur est représenté par le composant ressemblant à un transistor, le Pin central représente le signal, les Pin extérieurs sont l'alimentation.

Ici, on remarque que la résistance n'est pas oubliée !

```
#include "OneWire.h"
#include "DallasTemperature.h"

OneWire oneWire(A1);
DallasTemperature ds(&oneWire);

void setup() {
  Serial.begin(9600); // définition de l'ouverture du port série
  ds.begin(); // sonde activée
  pinMode(10, OUTPUT);
}

void loop() {
  ds.requestTemperatures();
  int t = ds.getTempCByIndex(0)

  Serial.print(t);
  Serial.println("C");

  if (t > 25) { digitalWrite(10, HIGH); }
  else { digitalWrite(10, LOW); }

  delay(100);
}
```

Figure IV.3.2 : Code Arduino

Voici le code en C++ (Figure IV.3.2) qui est utilisée pour le test et le fonctionnement du capteur sur notre Arduino. Il permet de retourner la valeur de température toutes les 100 ms dans ce cas-là.

Bien sûr ce « delay » sera largement augmenté lors de la version finale, ici ce n'est qu'un exemple qui montre le fonctionnement du capteur.

Nous avons rencontré quelques problèmes lors de la simulation du capteur de température, ce problème s'est d'abord trouvé dans le logiciel de simulation.

Le logiciel était le principal problème, car nous avons dû réinstaller le logiciel Arduino sur le poste de travail, car ce dernier a été erroné.

Ensuite, du téléchargement du logiciel, nous avons dû chercher les librairies utilisées pour pouvoir utiliser notre capteur de température et aussi de vérifier que notre programme respecte à la perfection l'ordre donné pour avoir une température correcte durant la fermentation de la bière.

#### o IV.4 – Conclusion + Analyse

Pour valider le montage ainsi que le code du capteur, nous avons dû faire subir à notre système plusieurs tests, tout simplement des tests comme :

→ mettre le capteur à l'air libre et obtenir un résultat qui doit correspondre au thermomètre numérique à  $\pm 0,5^\circ$

Ensuite, nous avons dû tester le temps de réponse du capteur après l'avoir laissé à l'air libre, car notre capteur devrait nous donner une valeur de température stable soit une température égale à la température extérieure d'environ  $12^\circ\text{C}$  au moment de notre test qui s'est déroulée pendant le mois d'octobre.

Dans le moniteur de série Arduino, le code nous renvoie comme prévu la valeur énoncée par le capteur, or cette valeur est renvoyé toutes les 100 ms par le code donc nous avons vérifier si le code ne transmettait pas des erreurs et qu'il respecte bien les incertitudes de température qui est de  $\pm 0,5^\circ\text{C}$ .

Cependant, nous avons constaté que cette valeur varie très rapidement lorsque l'on change d'endroit comme par exemple le passage de la salle de projet à la salle CA01.

Nous avons réalisé des tests avec le capteur, et nous avons constaté que pour passer d'un état à l'air libre à  $23^\circ\text{C}$  à un état dans l'eau à  $13^\circ\text{C}$  (mesuré grâce à un autre capteur) la valeur de température a mis 8,2 secondes à se stabiliser. Nous pouvons en conclure que le temps d'adaptation du capteur à son milieu ne sera pas un problème pour une mesure prévue une fois par heure.

## ➤ V – pH

### o V.1 – Introduction

Durant nos recherches sur les mesures intéressantes à la fermentation de la bière, la connaissance de l'acidité de la bière semblait être pertinente, nous avons commencé nos recherches afin de déterminer l'acidité d'un mélange.

### o V.2 – Fonctionnement

Nous avons donc commencé à étudier les différents capteurs de PH envisageables. Pour rappel, notre sonde PH doit remplir les critères suivants :

- Être intégrable dans notre boîtier
- Respecter une limite de tension ( 5V )
- Être compatible Arduino
- Résister à des températures très hautes et très basses
- Avoir une précision correcte
- Pouvoir résister à de longues périodes d'immersion
- Ne pas dépasser un certain prix

#### 1<sup>ère</sup> sonde : DFROBOT SEN0161

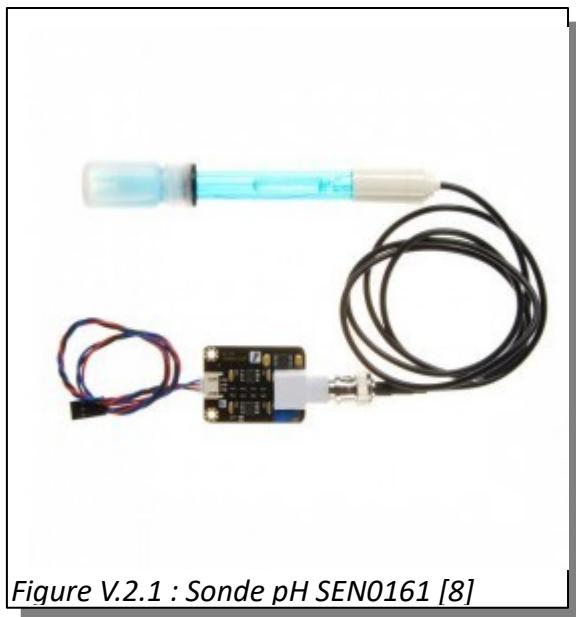


Figure V.2.1 : Sonde pH SEN0161 [8]

Cette sonde PH présente de nombreux avantages, car elle répond à tous nos critères sauf deux d'entre eux. Premièrement, l'embout de la sonde est très fragile et peut rendre celle-ci inutilisable lors de forts mouvements du liquide. Ensuite, l'intégration dans le boîtier est très compliquée, car il faut pouvoir faire passer l'embout de celle-ci à travers la paroi en plastique sans abîmer la sonde et le tout en gardant le trou étanche. Enfin, une note de bas de page nous indique que cette sonde est de qualité laboratoire, elle n'est donc pas adaptée pour une longue immersion. Nous ne pouvons donc pas choisir cette sonde.

## 2<sup>ème</sup> sonde : DFROBOT SEN0169



Figure V.2.2 : Sonde pH DFROBOT SEN0169 [9]

Cette sonde PH [9], cette fois-ci 2 fois plus chère, est une version pro de la précédente et est adaptée cette fois-ci à l'immersion longue durée. En revanche, elle présente les mêmes problèmes en termes de fragilité et d'intégration. Par ailleurs, même si cette sonde présente de meilleures caractéristiques techniques, cette différence rend l'embout de la sonde encore plus fragile. L'intégration est donc quasi-impossible et en fait donc une sonde inadaptée pour notre mesure.

### **o V.3 – Conclusion**

Finalement, les contraintes de fragilité et d'intégration autour de la sonde PH ne nous permettent pas d'assurer le bon fonctionnement de celle-ci dans toutes les situations. Après de nombreuses discussions avec nos professeurs, nous avons donc décidé d'écartier la mesure du PH.

## ➤ VI – Densité + Flottabilité

### ○ VI.1 – Introduction

Pour la densité, une donnée qui doit être mesuré et qui est revenu de nombreuses fois durant nos recherches se nomme la densité. La connaissance de celle-ci permet de mieux comprendre comment se déroule la fermentation au vu de sa consommation de sucre rendant la densité plus élevée.

L'objectif est donc de mesurer les variations de densité au sein de la bière en fermentation.

Pour cela, nous avions trois options :

- Tout d'abord, la solution de facilité est d'utiliser un capteur de densité ou un mustimètre électrique. Seulement ces capteurs sont hors de prix et trop encombrants. Cela ne représente donc pas une solution envisageable.
- Ensuite, nous pouvions déterminer la densité en utilisant la loi hydrostatique :  $dP/dz = -\rho g$ . Ainsi en connaissant la pression en deux points séparés d'une distance  $z$ , nous pouvons déterminer une variation de pression permettant d'obtenir une densité. Seulement, la variation de hauteur que nous pouvions obtenir était trop faible et les capteurs de pression auxquels nous pouvions avoir accès n'était pas assez précis au vu de cette petite hauteur rendant les erreurs trop grandes.
- Finalement, nous avons opté pour la solution d'utiliser un accéléromètre. Nous avons déterminé cette solution en faisant nos recherches sur le fonctionnement des appareils semblables au nôtre déjà sur le marché.

Suite à ces trois options, nous avons cherché par des schémas comment pouvons-nous faire pour que la boîte reste en surface dans la cuve et qu'elle soit immergée sans qu'elle ne puisse couler au fond de la cuve.

Pour cela, nos schémas vont être utilisés pour la partie de la densité et de la flottabilité, car nous avons ces recherches et ces manipulations en même temps.

Pour la flottabilité, nous avons remarqué que notre boîte doit flotter sur un côté pour qu'on puisse faire nos mesures de densité et de température, puis qu'on puisse garder la connexion avec la boîte, car la boîte devra nous renvoyer les informations qu'elle traitera.

Cette connexion sera expliquée dans la partie de la connexion entre la boîte et nous.

Pour obtenir un résultat sur la flottabilité de la boîte, nous avons dû poser des hypothèses pour pouvoir faire flotter la boîte dans les conditions que nous voulions avoir lors de l'expérimentation.

*Nous avons posé comme hypothèse :*

- que le poids sur le côté de la boîte 940 kg*
- que la boîte ne doit pas changer de position et qu'il doit rester dans la position verticale*

Suite aux trois choix retenues juste avant, nous avons tout d'abord trouvé qu'il est important de savoir que la conductivité électrique et la densité dans le contexte de la fermentation de la bière n'est pas directe. Cependant, certaines variations de la conductivité peuvent être associées à des changements dans la composition de la solution, y compris la concentration en ions. Pendant la fermentation, les sucres sont convertis en alcool et en dioxyde de carbone. Cela peut entraîner des changements dans la concentration d'ions dans la solution. Une augmentation de la conductivité peut être observée en raison de la présence accrue d'ions. Cependant, il est important aussi de noter que la conductivité seule ne permet pas une mesure précise de la densité. Des facteurs tels que la température et la composition chimique de la solution peuvent influencer la conductivité de manière complexe. Une calibration minutieuse serait nécessaire pour corrélérer les variations de conductivité avec les valeurs de densité spécifiques. Il serait donc judicieux de réaliser des tests préliminaires avec des échantillons de référence de densité connue et de calibrer le capteur pour tout type de bière pour

ainsi créer plusieurs corrélations en fonction du type de bière. Il faudrait alors préciser au programme Arduino quel type de bière est en cours de fabrication pour savoir quelle corrélation il devrait utiliser. Cependant, nous ne connaissons pas beaucoup de choses sur les types de bière et l'évolution de la conductivité pendant la fermentation de n'importe quel type de bière peuvent varier en raison de la diversité des recettes de bière et des conditions de fermentation. Il faudra donc se tourner vers une corrélation d'étalonnage le plus général possible. Cependant, il faut noter qu'il n'y a pas de norme universelle d'étalonnage pour la conductivité en fonction de la densité dans le contexte de la fermentation de la bière. En effet, la conductivité est un indicateur indirect et elle nécessite une calibration précise pour être fiable. Mais dans notre cas, on peut s'appuyer sur des données génériques associant la conductivité à la densité avec une certaine marge d'erreur relative variant de 5 à 10 %. Il est donc important de spécifier cette marge d'erreur pour l'utilisateur.

Voici un tableau associant la conductivité à la densité lors de la fermentation de la bière (à noter que ces valeurs n'ont pas de fondement empirique) :

Conductivité (micro-siemens/cm)	Densité (g/cm <sup>3</sup> )
500	1,040
1000	1,030
1500	1,020
2000	1,015
2500	1,010

Il est courant de viser une incertitude relative de l'ordre de 5 % à 10 % dans ce genre de situation. Par exemple, si la densité mesurée est de 1,020 g/cm<sup>3</sup>, une incertitude relative de 5 % pourrait signifier une fourchette de 1,019 à 1,021 g/cm<sup>3</sup>. Ce qui est plutôt acceptable.

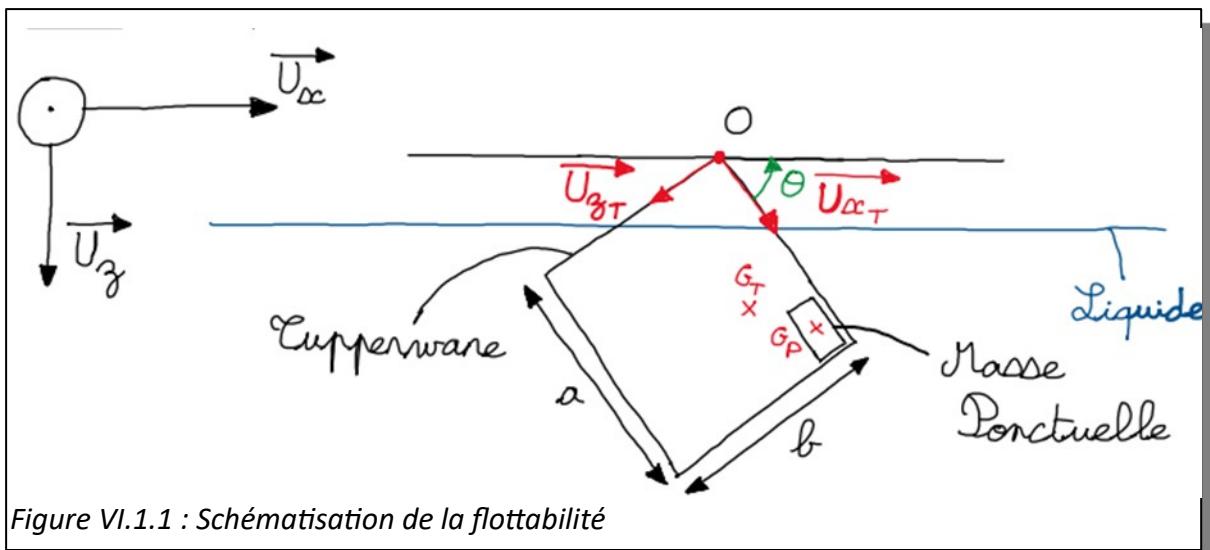


Figure VI.1.1 : Schématisation de la flottabilité

#### Description du schéma :

Le tupperware (Figure VI.1.1) flotte dans un liquide de densité  $\rho$ . Il est simplifié à un parallélépipède rectangle de hauteur  $h$ , de longueur  $a$ , de largeur  $b$  et de masse  $m_{tup}$ . On appelle  $V_{tup}$  son volume total et  $V_i$  son volume immergé. De plus, il possède une masse ponctuelle dans un coin de masse  $m_p$ . On positionne le repère orthonormé  $(\vec{u}_x, \vec{u}_y, \vec{u}_z)$  de centre 0. Ainsi que le repère  $(\vec{u}_{xT}, \vec{u}_{yT}, \vec{u}_{zT})$  tel que :

$$\begin{aligned}\vec{u}_{xT} &= \cos(\theta)\vec{u}_x + \sin(\theta)\vec{u}_z \\ \vec{u}_{zT} &= -\sin(\theta)\vec{u}_x + \cos(\theta)\vec{u}_z\end{aligned}$$

Avec  $\theta$  l'angle décrit entre les deux repères (soit du tupperware avec la surface de l'eau).

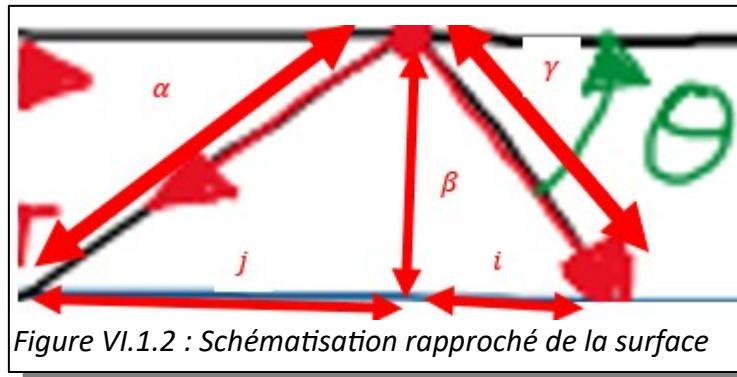
### Bilan des forces :

- Poids :
  - o Notée  $\vec{P}$
  - o Point d'application : centre d'inertie.
  - o De direction et sens  $\vec{u}_z$
  - o D'intensité  $\|\vec{P}\|=m_{tup}g$ ,  $g=\|\vec{g}\|$  étant l'accélération de la pesanteur.
- Poussé d'Archimède :
  - o Notée  $\vec{\Pi}$
  - o Point d'application : centre d'inertie du volume immergé du tupperware en considérant sa masse lors des calculs égal à  $\rho^*V$ .
  - o De direction et de sens  $-\vec{u}_z$ .
  - o D'intensité  $\|\vec{\Pi}\|=v_i\rho g$ .

### Centre de masse du Tupperware avec la masse ponctuelle :

$$\begin{aligned} \bar{x}_{GT} &= \int_0^a x dx \vec{u}_{xT} = \frac{a^2}{2m} (\cos(\theta) \vec{u}_x + \sin(\theta) \vec{u}_z) \\ z_{GT} &= \frac{1}{m} \int_0^b z dz \vec{u}_z = \frac{b^2}{2m} (-\sin(\theta) \vec{u}_x + \cos(\theta) \vec{u}_z) \\ \overrightarrow{OG} &= \bar{x}_{GT} + \bar{z}_{GT} = \frac{1}{2m} ((a^2 \cos(\theta) - b^2 \sin(\theta)) \vec{u}_x + (a^2 \sin(\theta) + b^2 \cos(\theta)) \vec{u}_z) \\ \frac{\overrightarrow{OG}_p}{m_G} &= a^2 (\cos(\theta) \vec{u}_x + \sin(\theta) \vec{u}_z) \\ \overrightarrow{OG} &= \overrightarrow{OG}_{T+p} = \frac{\overrightarrow{OG}_T \cdot m_{Tup} + \overrightarrow{OG}_p \cdot m_p}{m_{Tup} + m_p} \\ &\quad \left( \frac{1}{2} \Omega + a^2 \cos(\theta) \right) \vec{u}_x + \left( \frac{1}{2} \mu + b \sin(\theta) \right) \vec{u}_z \\ &\quad \frac{m_{Tup} + m_p}{m_{Tup} + m_p} \end{aligned}$$

### Calcul du centre d'inertie du volume immergé du Tupperware :



On retrouve ainsi :  $\alpha = \frac{\beta}{\sin(\theta)}$   $\gamma = \frac{\beta}{\cos(\theta)}$

Le centre de gravité (Figure VI.1.2) d'un triangle est égal à la moyenne des coordonnées des sommets :

$$\begin{aligned} m_s \cdot \vec{x}_{Gs} &= \frac{0+0+\alpha}{3} \vec{u}_{xT} = \frac{\beta}{3 \sin(\theta)} (\cos(\theta) \vec{u}_x + \sin(\theta) \vec{u}_z) \\ m_s \cdot z_{Gs} &= \frac{0+0+\gamma}{3} \vec{u}_{zT} = \frac{\beta}{3 \cos(\theta)} (-\sin(\theta) \vec{u}_x + \cos(\theta) \vec{u}_z) \\ m_s \cdot \vec{OG}_s &= \frac{\beta}{3} \left( \left( \frac{1}{\tan(\theta)} - \tan(\theta) \right) \vec{u}_x + 2 \vec{u}_z \right) \end{aligned}$$

On rappelle :  $\vec{OG} = \frac{\left( \frac{1}{2} \Omega + a^2 \cos(\theta) \right) \vec{u}_x + \left( \frac{1}{2} \mu + b \sin(\theta) \right) \vec{u}_z}{m_{Tup} + m_p}$

Ainsi,

$$\begin{aligned} \vec{OG}_{v_i} &= \frac{\vec{OG}_s \cdot m_s + \vec{OG}_T \cdot m_T}{m_s + m_T} \text{ avec } m_s + m_T = \rho V_i \\ &\quad \overbrace{\Omega'}^{\Omega'} \quad \overbrace{\mu'}^{\mu} \\ &\quad \frac{1}{2 \rho V_i} \left( \left( \Omega + \frac{2\beta}{3} \left( \frac{1}{\tan(\theta)} - \tan(\theta) \right) \right) \vec{u}_x + (\mu + 4) \vec{u}_z \right) \end{aligned}$$

#### Calcul des moments au point 0 :

$$\vec{M}_{\vec{p}} = \vec{OG} \wedge \vec{p} = \frac{1}{m_t + m_p} \begin{pmatrix} \frac{1}{2} \Omega + a^2 \cos(\theta) \\ 0 \\ \frac{1}{2} \mu + b \sin(\theta) \end{pmatrix} \wedge \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{2} \Omega + a^2 \cos(\theta) \\ 0 \end{pmatrix} \text{ avec } m = m_t + m_p$$

On obtient :  $\vec{M}_{\vec{p}} = -g \left( \frac{1}{2} \Omega + a^2 \cos(\theta) \right) \vec{u}_y$

$$\vec{M}_{\vec{p}} = \vec{OG} \wedge \vec{p} = \frac{1}{2 \rho V_i} \begin{pmatrix} \Omega' \\ 0 \\ \mu' \end{pmatrix} \wedge \begin{pmatrix} 0 \\ 0 \\ -\rho V_i g \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{2} \Omega' g \\ 0 \end{pmatrix} = \frac{g}{2} \Omega' \vec{u}_y$$

#### P.F.S :

$\sum \vec{F}_{ext} = \vec{0}$  projection sur z :  $m \cdot g = \rho V_i g$  avec  $V_i = abh - h \times \frac{1}{2} \beta (i+j)$  avec  $i = \frac{\beta}{\tan(\theta)}$  et  $j = \tan(\theta) \beta$

$$m = \rho \left( abh - \frac{h \beta^2}{2} \left( \frac{1}{\tan(\theta)} - \tan(\theta) \right) \right)$$

$\sum \vec{M} = \vec{0}$  projection sur y :  $\Omega' = 2 \Omega$

$$\Omega + \frac{2\beta}{3} \left( \frac{1}{\tan(\theta)} - \tan(\theta) \right) = \Omega + 2a^2 \cos(\theta)$$

$$\beta = \frac{3a^2 \sin(\theta) \cos^2(\theta)}{\cos(2\theta)}$$

On remplace  $\beta$  dans l'équation précédente :

$$m = \rho \left( ab h - h \left( \frac{3a^2 \sin(\theta) \cos^2(\theta)}{\cos(2\theta)} \right)^2 \left( \frac{1}{\tan(\theta)} - \tan(\theta) \right) \right)$$

$$m = \frac{\rho h (ab - 9a^2 \sin(\theta) \cos(\theta)^3)}{2 \cos(2\theta)}$$

On isole  $\rho$  :

$$\rho = \frac{m 2 \cos(2\theta)}{h (ab - 9a^2 \sin(\theta) \cos(\theta)^3)}$$

## o VI.2 – Fonctionnement

Le principe de fonctionnement s'appuie sur la rotation du boîtier dû au changement de densité. En effet, avec une répartition de masse réfléchie, lorsque la densité augmente seulement une partie du boîtier se déplace et pivote autour de la partie qui flotte déjà. À l'aide d'un accéléromètre, on peut déterminer la rotation qu'a eu le boîtier dû à la variation de densité. Finalement, en établissant une relation empirique entre ces deux variations, on peut établir une relation entre elles.

Suite aux exigences données par les différentes analyses du système, on doit pouvoir faire des mesures avec le Tupperware non-vertical, mais penchant sur un côté.

On remarque sur les photos ci-dessous que le Tupperware doit se retrouver sur cette position pour permettre aux capteurs de faire le travail dans de bonnes conditions.



Figure VI.2.1 : Test de la Flottabilité de la boîte

Suite à cette expérimentation, nous avons passé de l'hypothèse que le Tupperware devait être incliné sur un angle, et nous avons choisi d'incliner le Tupperware sur un côté vue sur les photos ci-dessous pendant l'expérimentation de la flottabilité du Tupperware.

Lors de cette flottabilité, nous avons cherché jusqu'à quelle limitée de poids, le Tupperware pouvait rester en flottabilité dans l'eau, or le Tupperware ne sera pas en flottabilité dans l'eau, mais dans la bière.

Mais nous avons posé comme hypothèse lors de cette expérimentation que la bière était assimilée à de l'eau.

Donc nous avons pu trouver comme limite de poids maximal du Tupperware pour sa flottabilité :



Figure VI.2.2 : Poids de la boîte plein

Or, nous avons fait la mesure du poids du Tupperware chargé jusqu'à sa limite de flottabilité, or pour connaître le poids maximal qui permettrait au Tupperware de rester dans sa phase de flottabilité.

Nous devons connaître le poids du Tupperware vide, or, nous pouvons connaître ce poids en mesurant le Tupperware à vide, nous pouvons l'observer sur la photo ci-dessous :



Figure VI.2.3 : Poids de la boîte vide

Après avoir eux le poids du Tupperware à vide (Figure VI.2.3) et chargé (Figure VI.2.2), nous pouvons en déduire du poids de charge que nous devrions mettre dans le Tupperware pour rester en flottabilité lors de nos mesures avec les différents appareils de mesure.

Le poids maximal de la charge vaut 936,7g.

Or, nous devrions faire attention que le poids maximal des appareils de mesure ne doit pas dépasser cette valeur du poids maximal de la charge.

Si le poids maximal est dépassé, nous devrions trouver une autre solution pour que le Tupperware reste dans la phase de flottabilité, comme par exemple mettre une bouée qui permettrait au Tupperware de rester dans cette phase de flottabilité.

Or, pour vérifier que notre vérification manipulation soit correcte, nous devrions faire une analyse expérimentale par des schémas et des calculs pour que le Tupperware reste bien dans la phase de flottabilité.

## ➤ VII – Batterie + Antenne Relais

Pour la batterie, nous avons constaté que cela était l'une des difficultés majeures liée à notre système qui consistait à en choisir une qui serait capable d'alimenter notre système le plus longtemps possible avec le fonctionnement par intermittence grâce à l'ESP32 (car la fermentation de la bière dure plusieurs semaines) et capable d'être rechargeée.

La première idée qui nous est venue consistait à mettre des piles en séries, ce qui obligerais l'utilisateur à les changer souvent et d'avoir une réserve de piles à sa disposition, ce qui de plus représentait un certain budget à dépenser.

Cependant, il gagnerait en rapidité, car il lui suffirait de changer les piles pour remettre le système en place dans la solution de fermentation.

Enfin, en fonction des piles, la durabilité de l'autonomie du système n'était pas forcément garantie.

L'autre solution que nous avons choisie et vers laquelle nous nous sommes tourné consistait à utiliser une batterie externe avec deux ports externes dont le OUTPUT servirait à alimenter le système (branchement possible avec la carte Arduino) et le INPUT à le recharger avec un câble de type USB.

L'autonomie serait garantie pendant longtemps et la recharge ne durerait pas plus de deux heures.

Il sera donc important de choisir une batterie externe capable de durer dans le temps par intermittence et donc de tester la batterie sur une longue durée.

```
void loop()
{
    // DeepSleep
    unsigned int sleepCounter;
    for (sleepCounter = 7.5; sleepCounter > 0; sleepCounter--)
    {
        LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
    }
}
```

Figure VII.1 : Bibliothèque DeepSleep

Pour la partie de l'antenne-relais, nous avons constaté que notre projet devait utiliser la bibliothèque DeepSleep pour Arduino et cette utilisation nous a permis de mettre la carte en veille pour une durée spécifique que nous avons déterminée. Les tests ont montré une consommation d'énergie presque négligeable, ce qui prolonge considérablement la durée de vie de la batterie. Cependant, la carte ne peut se mettre en veille que cependant 8 secondes avec son horloge interne. Pour prolonger cette durée, nous envisageons de placer cette horloge dans une boucle afin d'obtenir des périodes de veille plus longues. Pour tester l'efficacité de ce mode, nous avons mesuré la consommation des capteurs grâce à un ampèremètre avec et sans DeepSleep. On peut remarquer la bibliothèque de DeepSleep sur la Figure VII.1.

```
// Capteur de turbidité
int sensorValue = analogRead(A0); // read the input on analog pin 0;
float voltage = sensorValue * (5.0 / 1024.0); // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V);
```

Figure VII.2 : Configuration du capteur de turbidité

Nous avons vérifié avec plusieurs tests que le code de la Figure VII.2 permettaient de mesurer la valeur sur l'entrée analogique A0 et en convertissant l'analogie en valeur de voltage, et ceci respectait ces conditions à la perfection.

```
// Capteur de température
float temperature = getTemp();
Serial.println(temperature);
delay(100); //just here to slow down the output so it is easier to read
```

Figure VII.3 : Configuration du capteur de température

Nous avons fait plusieurs tests sur le code de la Figure VII.3 qui devraient permettre de mesurer la température en la convertissant, mais nous n'avons pas fini les vérifications de ce code, car nous avons oublié quelques paramètres majeurs à ces tests.

Nous avons réalisé le code Arduino complet de l'antenne-relais, or celui-ci n'est pas tout à fait fini, car nous avons remarqué quelques erreurs lors de nos nombreux tests (Figure 4, Figure 5 et Figure 6).

Nous avons constaté que lors de nos nombreux tests sur l'antenne-relais, nous avons branché les différents paramètres comme sur la Figure VII.4.

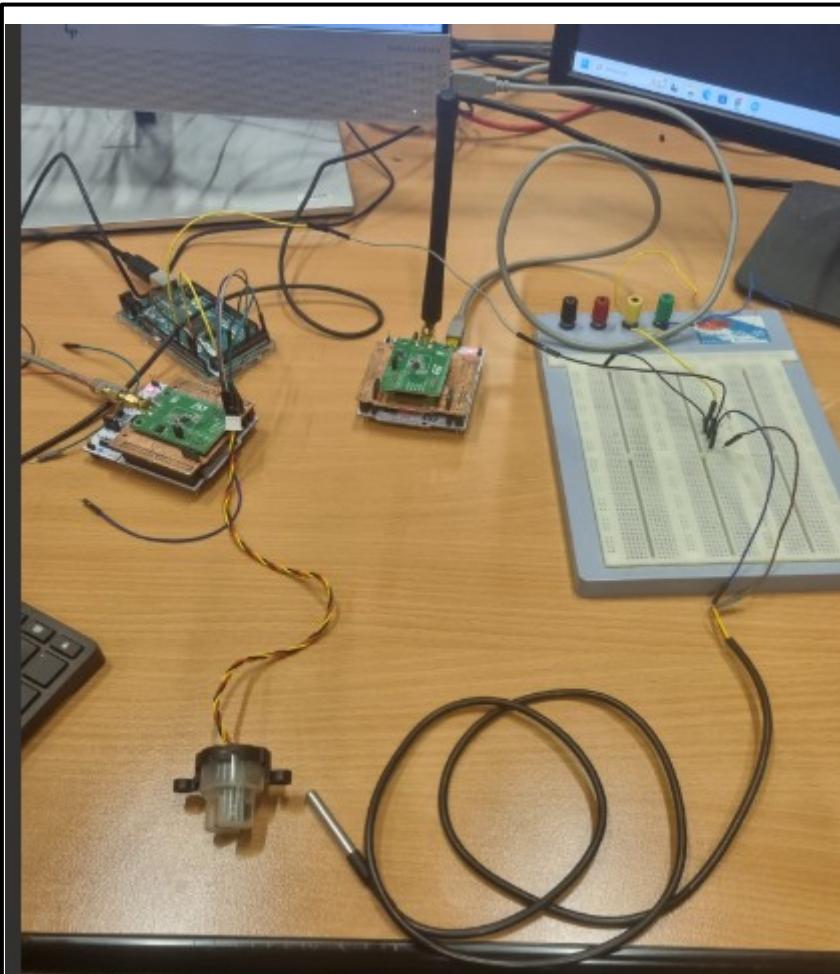


Figure VII.4 : Câblage de l'antenne relais

Cependant, après la réception des données par l'antenne-relais, nous avons cherché une méthode supplémentaire, nous permettons de faire des analyses plus précises des mesures faites durant la fermentation.

Dans ce projet, l'utilisation de la bibliothèque DeepSleep (Figure VII.4) pour Arduino nous permet de mettre la carte en veille pour une durée spécifique que nous avons déterminée. Les tests ont montré une consommation d'énergie presque négligeable, ce qui prolonge considérablement la durée de vie de la batterie. Cependant, la carte ne peut se mettre en veille que pendant 8 secondes avec son horloge interne. Pour prolonger cette durée, nous envisageons de placer cette horloge dans une boucle afin d'obtenir des périodes de veille plus longues.

Pour tester l'efficacité de ce mode nous avons mesuré la consommation des capteurs grâce à un ampèremètre avec et sans DeepSleep

Pour pouvoir donner du sens à notre projet et le rendre le plus réaliste possible, il faut pouvoir les visualiser et éventuellement les manipuler. Cette approche permet de simuler un accès de la part de l'utilisateur sur ses données. Simplement, il faut pouvoir envoyer les données depuis notre carte Arduino et les récupérer sur un autre support tel qu'une Raspberry pi et pouvoir les visualiser, mais par quels moyens ?

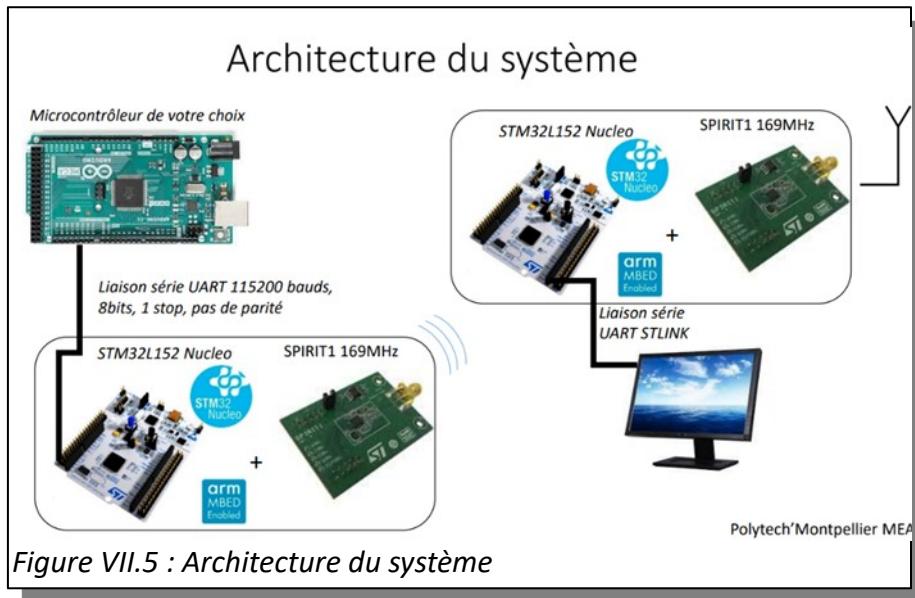
### La partie émission :

La partie émission sera composée dans notre projet d'un émetteur RF (radiofréquence) sur une carte STM32 qui sera branché à notre carte Arduino. Mais qu'est-ce qu'un émetteur et comment cela fonctionne-t-il ?

Un émetteur RF fonctionne comme un « talkie-walkie », une onde électrique est modulée et amplifiée pour être projeté dans l'air jusqu'à atteindre une antenne réceptrice. Dans notre projet, nous utiliserons l'émetteur RF SPIRIT1 qui envoie le signal à 169MHz.

### La partie réception :

La partie réception est similaire à l'émission avec également un SPIRIT1 169MHz, mais cette fois-ci accompagnée d'une antenne-relais pour recevoir les ondes transmises.



La liaison série UART sur la Figure VII.5 permettent la communication entre l'Arduino et l'émetteur. Elle est assurée par un débit en bauds qui est le nombre de fois par seconde où le signal de communication série change d'état (fréquence, tension etc...) : 115 200.

Maintenant, que nous savons comment fonctionne notre communication, nous allons nous intéresser au contenu de notre data. On souhaite récupérer les valeurs de plusieurs paramètres

(température, turbidité et accéléromètre) et pour cela, la mise en forme en trame JSON est bien adaptée pour notre cas.

- Le message JSON est une chaîne de caractère qui doit respecter le format suivant :

```
{"ID":1,"Temp":23.4,"Pression":1,"Dens":56}
```

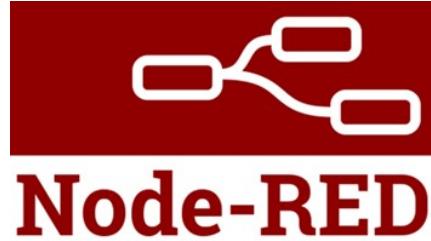
Numéro de groupe de 1 à 5      Valeurs à remplacer par celles de vos capteurs      A remplacer par le nom des paramètres mesurés par vos capteurs

- Cette trame doit être envoyée via une liaison série UART à 115200 bauds, 8bits, 1 stop, pas de parité

*Figure VII.6 : Décomposition de la trame*

Mais, sur la Figure VII.6, ces données doivent bien être stockés quelque part et n'oublions pas notre objectif de permettre à l'utilisateur de les visualiser. Nous avons donc plusieurs voies possibles :

#### Node-RED :



Node-RED est un outil de programmation visuelle open-source qui simplifie la création d'automatismes et de flux de données en utilisant un environnement graphique basé sur des nœuds interconnectés. Avec une interface intuitive, les utilisateurs peuvent glisser-déposer des nœuds représentant des actions ou des traitements, puis les connecter pour créer des flux d'exécution. Principalement utilisé dans l'Internet des objets (IoT), Node-RED facilite la création d'applications et d'automatisations en permettant aux développeurs et aux amateurs de connecter divers dispositifs et services avec une approche visuelle et accessibles, nécessitant peu ou pas de compétences en programmation traditionnelle.

#### Grafana :



Grafana est un outil de visualisation de données open-source qui offre une interface conviviale pour créer des tableaux de bord interactifs et informatifs. Principalement utilisé pour surveiller et analyser des données provenant de diverses sources telles que des bases de données, des systèmes de surveillance, ou des capteurs IoT, Grafana permet aux utilisateurs de créer des graphiques dynamiques, des diagrammes et des alertes. Son support pour de nombreuses sources de données et son extensibilité en font un choix populaire pour la visualisation de données dans des domaines tels que l'analyse des performances du système, la surveillance des applications, et la gestion des opérations.

## **InfluxDB :**



InfluxDB est une base de données de séries chronologiques open-source conçue pour stocker, interroger et récupérer efficacement des données temporelles. Spécialement adaptée pour gérer des flux continus d'informations générées par des capteurs, des systèmes de surveillance, ou des applications IoT, InfluxDB excelle dans la collecte et l'analyse de données qui évoluent avec le temps. Grâce à son modèle de données optimisé pour les séries chronologiques et sa simplicité d'utilisation, InfluxDB est largement utilisé pour stocker et récupérer des données temporelles à des fins de monitoring, de reporting, et d'analyse.

En résumé, nous avons Node-RED qui permet la programmation d'automatismes de flux de données, Grafana qui permet la visualisation de ces données et enfin InfluxDB qui est une réunion des deux. De plus, il existe une bibliothèque InfluxDB sur python qui facilite la gestion de notre base de données. À noter qu'InfluxDB est installé sur notre Raspberry pi en tant que serveur.

Finalement, InfluxDB est l'outil qui est le plus adapté à notre future utilisation. On va pouvoir récupérer nos données en python sur notre Raspberry pi pour les traiter et les envoyer sur notre database.

On établit tout d'abord la liaison entre l'émetteur et la carte Arduino. Pour cela, on a juste à connecter l'émetteur et la carte ensemble puis le code final de notre Arduino dispose d'une partie spécifiant la liaison (Figure VII.7).

## DEMO : Envoie de données capteurs

- Envoie de données issues des entrées analogique A0, A1, A2 d'une arduino Mega
- Réception par le concentrateur SPIRIT1 avec affichage sur une console windows

```
int sensorValueA0 = 0;
int sensorValueA1 = 0;
int sensorValueA2 = 0;

void setup()
{
    Serial.begin(115200);
    Serial1.begin(115200);
    Serial.println("init...");
}

void loop()
{
    sensorValueA0 = analogRead(A0);
    sensorValueA1 = analogRead(A1);
    sensorValueA2 = analogRead(A2);

    Serial1.print("ID:1,A0:");
    Serial1.print(sensorValueA0);
    Serial1.print(",");
    Serial1.print("A1:");
    Serial1.print(sensorValueA1);
    Serial1.print(",");
    Serial1.print("A2:");
    Serial1.print(sensorValueA2);
    Serial1.println(");

    delay(3000);
}
```



The image shows a breadboard setup with an Arduino Uno and a SPIRIT1 module. Wires connect the Arduino pins to the breadboard, which then connects to the SPIRIT1 module. A laptop is connected to the SPIRIT1 module via a USB cable.

```
rss = -50.0
{ID:1, MagX:-3674, MagY:3894, MagZ:6279, Temp:34}
rss = -50.5
{ID:1, MagX:-3663, MagY:3894, MagZ:6275, Temp:33}
rss = -50.5
{ID:1, MagX:-3667, MagY:3884, MagZ:6284, Temp:33}
rss = -51.0
{ID:1, MagX:-3670, MagY:3884, MagZ:6280, Temp:33}
rss = -53.0
{ID:1, MagX:-3672, MagY:3906, MagZ:6274, Temp:34}
Polytech'Montpellier ME
```

The text on the right side of the image shows the output of the serial communication, displaying RSSI values and sensor data for three channels (A0, A1, A2) over time.

Figure VII.7 : Démo de l'envoi des données

Maintenant, concentrons-nous sur la réception des données, c'est-à-dire la partie antenne relais et Raspberry pi. On identifie d'abord le chemin d'accès au fichier périphérique du récepteur connecté à la Raspberry pi en lançant la commande suivante dans le terminal sur la Figure VII.8 :

```

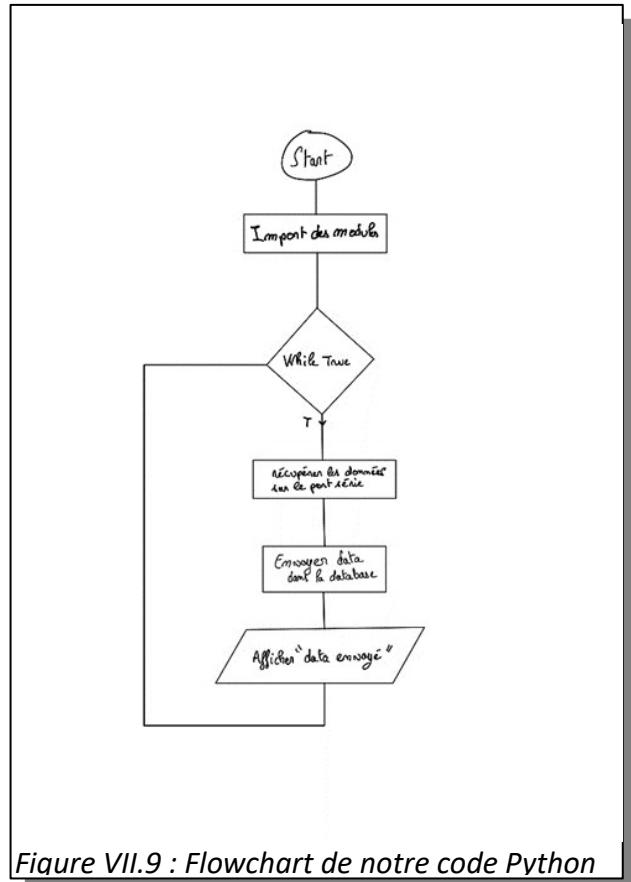
bilel@debian: /dev$ ls
autofs          rtc           ttyIf        tty38  tty4f  vcsf
block          log           tty1f        tty39  tty6e  vcs
bsg            loop          tty17        tty39  tty66  vcs
btrfs-control   loop2         tty18        tty40  tty67  vcs
bus             loop3         tty19        tty41  tty68  vcs
cdrom          loop4         tty20        tty42  tty69  vcs
char            loop5         tty21        tty43  tty70  vcs
console         loop6         tty22        tty44  tty71  vcs
core            loop7         tty23        tty45  tty72  vcs
cpu_dma_latency loop-control  snd          tty24        tty46  vcs
cuse            mapper       stderr        tty25        tty47  vcs
disk            mqueue      stdin         tty26        tty48  vcs
dri             net          stdout        tty27        tty49  vhid
fd              null         tty          tty28        tty50  vinput
full            nvram        tty          tty29        tty51  urandom
fuse            port         tty          tty30        tty52  vboxguest
hidraw          ppp          tty          tty31        tty53  vboxuser
hpet            psaux        tty          tty32        tty54  vcs
hugepages       ptmx         tty          tty33        tty55  vcs1
hwmon          pts          tty          tty34        tty56  vcs2
initctl         random       tty          tty35        tty57  vcs3
                         tty          tty36        tty58  vcs4
                         tty          tty37        tty59  vhost-net
                         tty          tty38        tty60  vhost-vsock

```

Figure VII.8 : Réception des données

Une connexion/déconnexion nous permet d'identifier notre récepteur.

Nous avons maintenant tous les éléments pour pouvoir construire notre code final. Pour une visualisation en temps réel, il faut que notre data s'envoie en continu et donc que notre fichier python soit constamment en exécution.



*Figure VII.9 : Flowchart de notre code Python*

### **Flowchart du code final**

Après avoir fait des corrections et des ajustements (Figure VII.9), nous avons le code final suivant :

```

import time
import random
from datetime import datetime
import json

url = 'http://162.38.110.107:8086'
token = '5wSNB8AMMK6pV2X_PXTHZCJSMgAEvaDzbm5JNsINIITAJU2wv1_DRP906odmvFLSxpOMN-xylnp2KqBscTfDXw=='
bucket = 'TEST_F'
org = 'MEA3'

client = InfluxDBClient(url=url, token=token)

data = {
    "ID" : 4,
    "A0" : "Valeur0",
    "A1" : "Valeur1",
    "A2" : "Valeur2"
}

i = 1

kind0 = 'turbidité'
kind1 = 'température'
kind2 = 'accéléromètre'
host = 'host1'
device = 'Arduino'

while True :

    data["A0"] = random.uniform(0.5,5.0)
    data["A1"] = random.uniform(0,40.0)
    data["A2"] = random.uniform(-4.0,4.0)
    #data = port.readline() #on lit le port série

    today = datetime.now() #ajout de la date en ISO8601
    iso = today.isoformat()

    data.update({"time" : str(iso)})

    #A0
    point = Point(kind0).tag('host', host).tag('device', device).field('value', data["A0"]).time(time=datetime.utcnow())
    write_api = client.write_api(write_options=SYNCHRONOUS)
    write_api.write(bucket = bucket, org = org, record=point)

    #A1
    point = Point(kind1).tag('host', host).tag('device', device).field('value', data["A1"]).time(time=datetime.utcnow())
    write_api = client.write_api(write_options=SYNCHRONOUS)
    write_api.write(bucket = bucket, org = org, record=point)

    #A2
    point = Point(kind2).tag('host', host).tag('device', device).field('value', data["A2"]).time(time=datetime.utcnow())
    write_api = client.write_api(write_options=SYNCHRONOUS)
    write_api.write(bucket = bucket, org = org, record=point)

    print("data ",i," envoyé\n")

    i += 1
    time.sleep(3)

```

Figure VII.10 : Code Python

### Code final :

Les données maintenant importées dans notre database, nous pouvons créer un Dashboard avec différents graphes par paramètre pour une meilleure compréhension et analyse des données (Figure VII.10). Les graphes nous montrent les dernières valeurs depuis un certain temps (ici depuis les 5 dernières minutes). Nous pouvons aussi fixer une actualisation automatique pour mieux remarquer les changements de valeurs (Figure VII.11).



Figure VII.11 : Simulation de la température

On peut également interroger notre base de données selon les paramètres voulus (Figure VII.12). Ici, nous avons demandé de représenter sous forme de graphe les valeurs de l'accéléromètre le 16 janvier de 16h45 à 17h.



Figure VII.12 : Différentes simulations répondant au cahier des charges

L'utilisateur dispose d'énormément de possibilités grâce aux capacités d>InfluxDB que ce soit en termes de stockage, d'interrogation et de visualisation de données. Finalement, le client disposera de codes d'accès à InfluxDB pour visualiser très simplement ses données et adapter sa production en conséquence.

## ➤ Mode d'emploi

Boîtier de mesure de la température, turbidité et densité d'une solution de fermentation de la bière.

### Tables des matières :

- Liste des composants
- Lecture des données mesurées
- Les étapes de l'emploi du boîtier
- Conseils et astuces

### Liste des composants :

Le tupperware de 1L est composé des éléments suivants :

- 1 capteur de turbidité ( réf : SEN0189 )
- 1 sonde de température ( réf : DS18B20 )
- 1 module accéléromètre gyroscope ( réf : MPU6050 )
- 1 carte Arduino Méga ( réf : 2560 )
- 1 résistance de 4.7 kΩ
- 1 batterie externe de 5000 mAh
- 1 module de communication de données branché sur STM32

### Lecture des données mesurées :

Assurez-vous de disposer de tous les composants requis :

- L'antenne-relais branchée sur STM32
- Une carte Raspberry pi
- Un ordinateur

Dans un premier temps, il vous faudra saisir le code d'accès sur la plateforme influxdb.com pour visualiser les données mesurées par le boîtier. C'est une plateforme qui permet de stocker des données en continu dans le temps. Son atout, c'est de pouvoir récupérer les données stockées à tout moment. Il est également possible de prendre toutes les données sur une période pour les visualiser sous forme de graphe(s) par exemple. Il est fortement recommandé de faire tourner ce programme sur un serveur pour une application de type industrielle. Il est également possible de télécharger ces données mesurées pour les réutiliser ailleurs. Pour plus d'informations, veuillez consulter le site officiel : InfluxDB OSS v2 Documentation (influxdata.com) .

Pour résumer, l'antenne de transmission contenue dans le boîtier transmet les caractéristiques mesurées vers l'antenne de réception branchée sur STM32 qui transmet les données à une carte Rasberry pi dont le code python traite les informations pour les stocker sur influxdb.

### **Les étapes de l'emploi du boîtier :**

- 1 – Dévisser les dix vis sur le couvercle avec un tournevis approprié
- 2 – Appuyer sur le bouton « on » de la batterie pour alimenter le système. Ne rien toucher d'autre ! Attention également au câble.
- 3 – Avant de refermer le couvercle, vérifier que tout marche bien et que les données sont bien transmises à l'antenne et qu'on visualise bien les mesures des caractéristiques sur l'écran.
- 4 – Refermer le boîtier et visser les vis. Attention à ne pas trop forcer lorsque vous vissez pour éviter de casser le couvercle.
- 5 – Plonger l'objet dans une solution dans la position verticale.

### **Conseils et astuces :**

- Pensez à nettoyer l'extérieur du boîtier après et avant chaque utilisation avec des produits adéquats.
- Il ne faut pas oublier de recharger la batterie régulièrement, même pendant la phase de mesure des caractéristiques lors de la fermentation.
- En cas de panne, contactez le 03 14 15 92 65 pour obtenir de l'aide.
- Pensez à vérifier que les vis sont correctement vissées et que le boîtier reste toujours étanche avant de le plonger dans la solution de fermentation.
- Il vous faudra munir d'un câble USB ou USBC pour recharger la batterie à l'intérieur du boîtier.
- Pensez à éteindre la batterie à la fin de l'utilisation.
- Adaptez l'interface d'affichage sur influxdb pour un usage plus confortable.

## **➤ Retour d'expérience**

Durant ce projet, le groupe a rencontré quelques problèmes lors de l'organisation des parties, nous permettant de répondre au cahier des charges. Mais en grande partie, les problèmes ont été rencontrés la compréhension de la conception de notre prototype pour le projet et la recherche des différentes solutions qui seront utilisées pour répondre au cahier des charges.

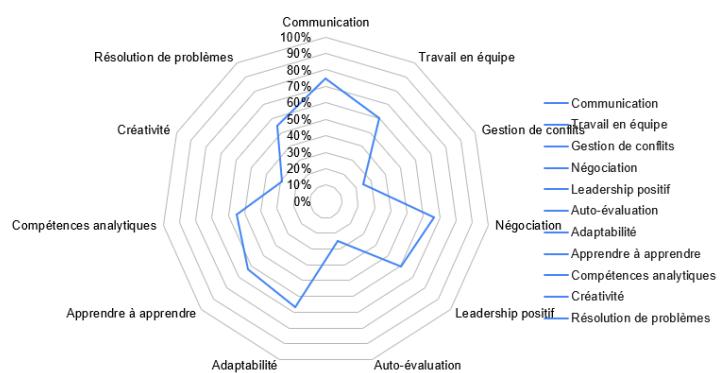
Nous avons pu constater que ce projet a pu été certain d'entre nous sur la compréhension du travail en groupe et sur le rôle d'un ingénieur qui ne nous est pas entièrement assimilé avant d'entre en école d'ingénieur.

**Pierre BOUCKSON**



## Compétences

Mes compétences - Synthèse



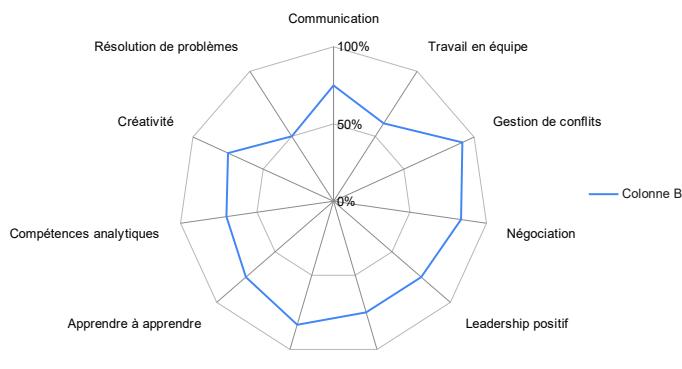
Lors de ce projet, Pierre a pu montrer au groupe son incroyable adaptation, dont on pouvait solliciter sur la compréhension du projet, mais aussi sur son implication dans le projet.

**Théo FOULQUIER**



## Compétences

Mes compétences - Synthèse



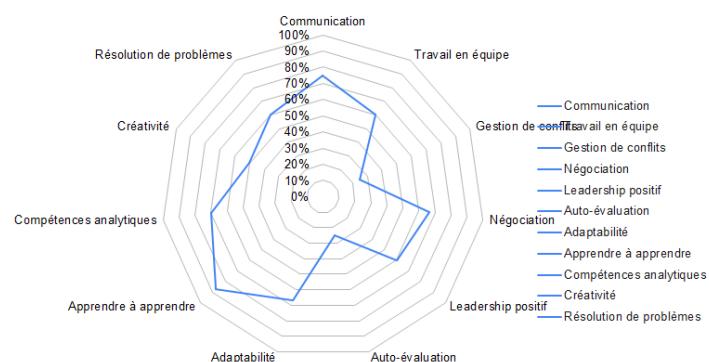
Théo a montré au groupe un rôle de leadership très convaincant et dont sans lui le groupe ne pouvait pas rester dans le droit chemin.

**Khaoula LAHBIB**



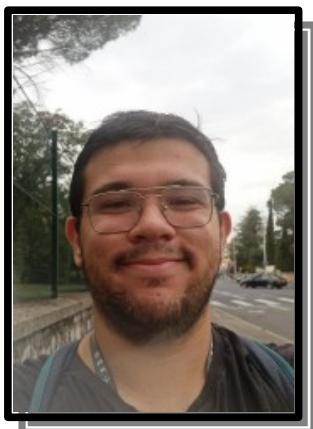
## Compétences

### Mes compétences - Synthèse



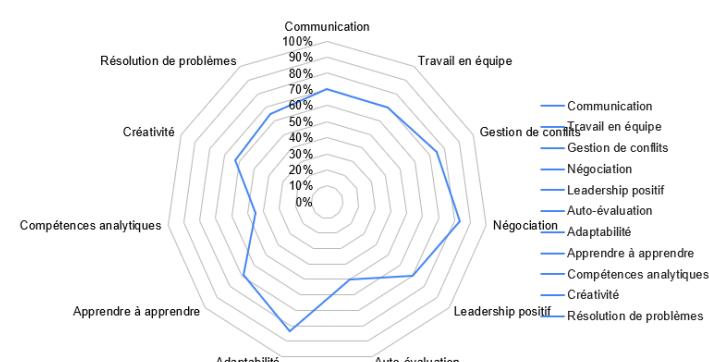
Khaoula a permis au groupe de comprendre comment fonctionner un projet en groupe, car étant une redoublante elle nous a expliqué comment organiser notre travail, et comment bien communiquer entre nous.

**Mathieu RICHARD**



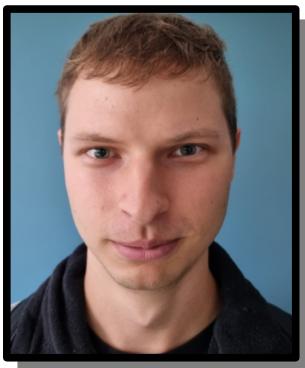
## Compétences

### Mes compétences - Synthèse



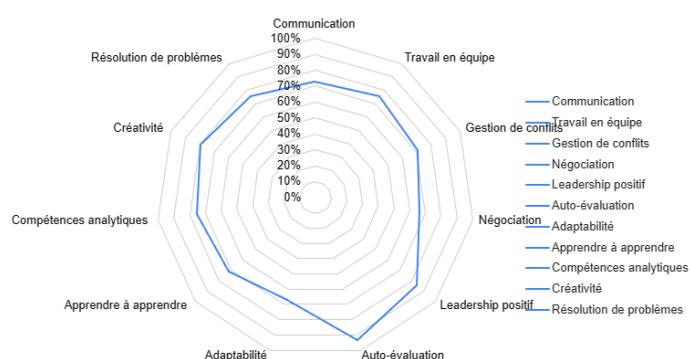
Mathieu a démontré une incroyable implication dans le projet par une très grande créativité, mais aussi dans la résolution des problèmes.

**Edgars VASILJEVS**



**Compétences**

Mes compétences - Synthèse



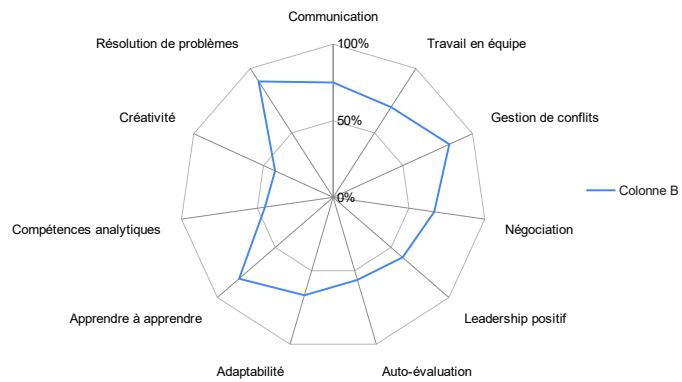
Edgars a joué un rôle important dans la communication du groupe, car il a pu aider le groupe quand nous avons rencontré des problèmes.

**Maxime VITAL**



**Compétences**

Mes compétences - Synthèse



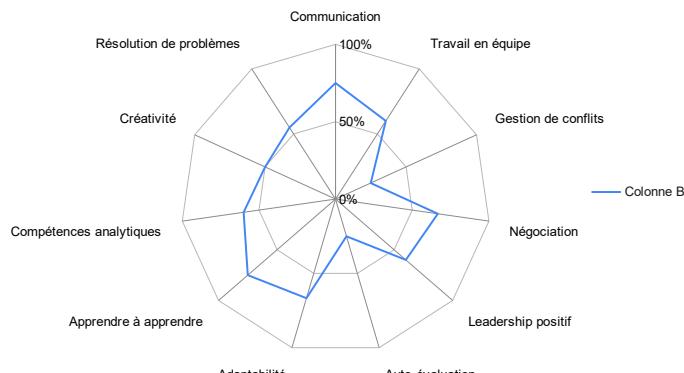
Maxime a su montré une très organisation du travail, car sans lui et sans l'utilisation du logiciel Beesbusy, le groupe n'aurait pas pu respecter les différentes dates limite du projet pour la remise du rapport final, l'oral de projet et la remise de chaque rapport hebdomadaire.

**Ambre VUAILLAT**



**Compétences**

**Mes compétences - Synthèse**



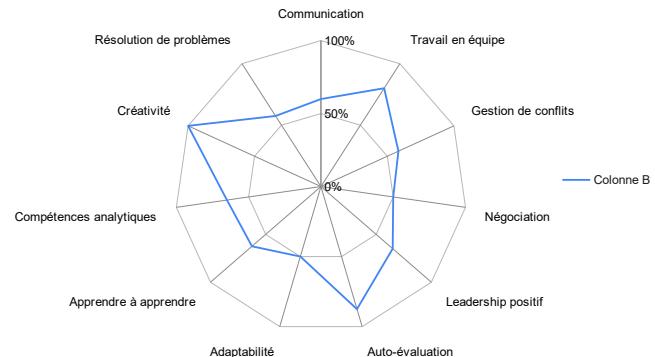
Ambre a su aider le groupe dans la communication, et par cette communication, elle a pu comprendre le rôle de chacun dans le groupe et elle est à aider à communiquer entre eux.

**Bilel ZAKANI-FADILI**



**Compétences**

**Mes compétences - Synthèse**



Bilel a, durant ce projet, montré une très implication et plus précisément dans la communication entre la boîte et le l'ordinateur qui recevra les données, car nous avons su que la communication entre la boîte et le fabricant serait difficile. Il a su apprendre dans les délais impartis cette communication et savoir l'utiliser.

## ➤ Conclusion

Lors de ce projet, nous n'avons pas eu le temps de tester notre prototype dans une solution qu'on aura assimilé à la bière, qui nous aurait permis de faire les différentes mesures avec nos capteurs de température, notre sonde pH et notre accéléromètre.

Nous avons pu vérifier l'étanchéité de notre boîte sur les Figure 7 et 8, et nous avons pu assembler tous les capteurs et vérifier s'ils fonctionnaient et qui réalisent bien les différents codes Arduino introduits à chacun (Figure 9).

## ➤ Annexe

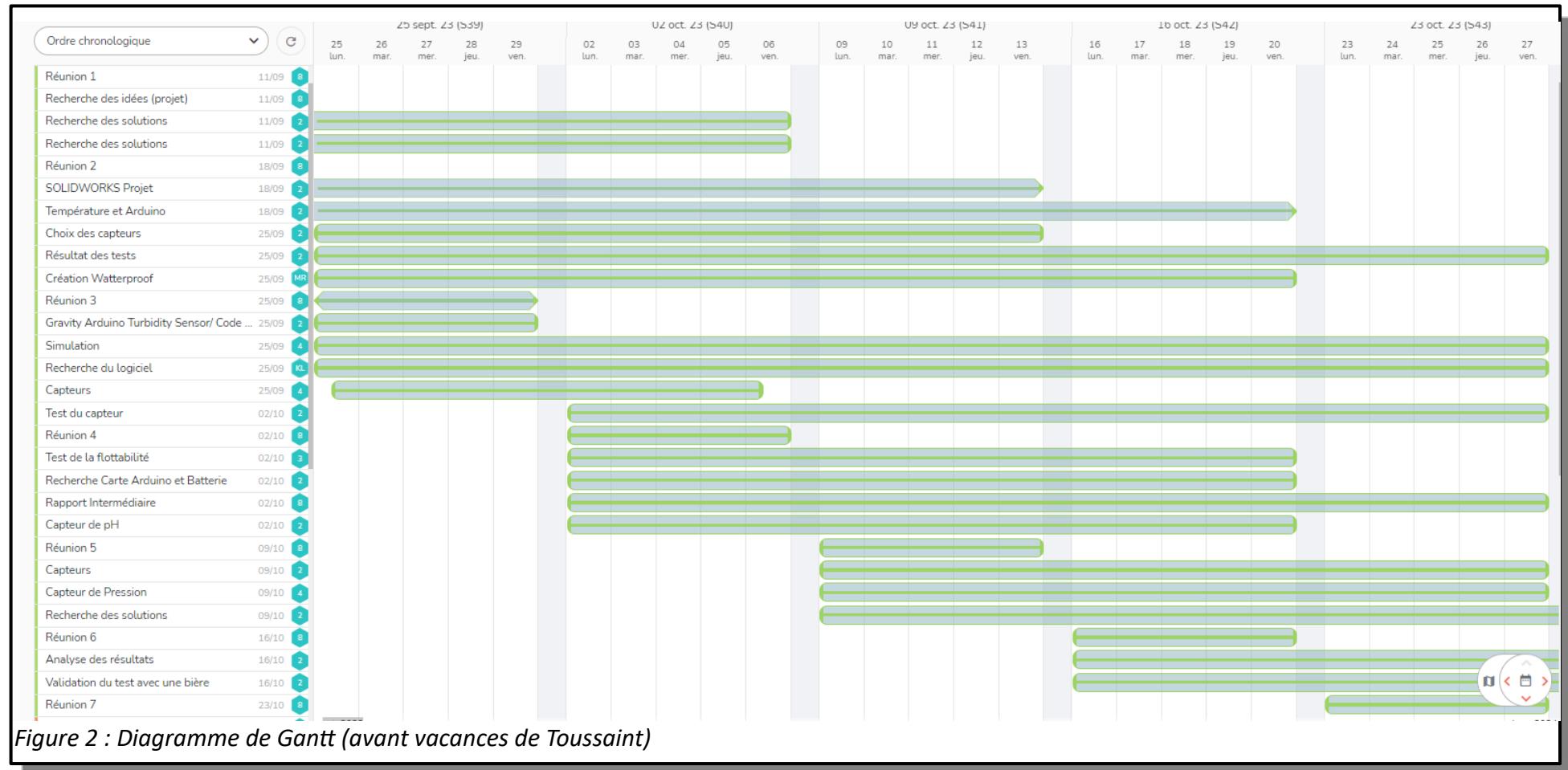
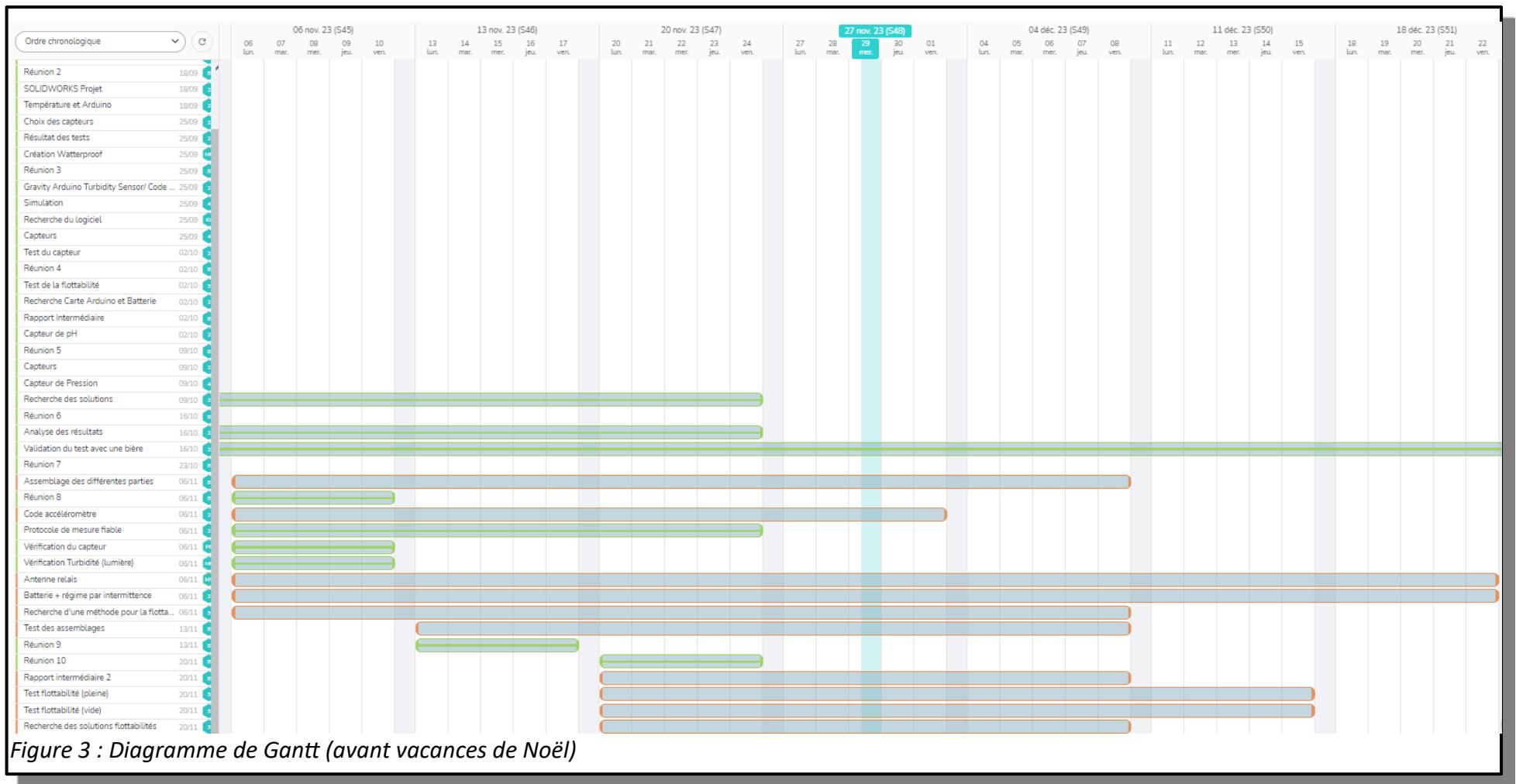


Figure 2 : Diagramme de Gantt (avant vacances de Toussaint)



```

#include <OneWire.h>
int sensorValueA0 = 0;
int sensorValueA1 = 0;
int sensorValueA2 = 0;
int DS18S20_Pin = 2; //DS18S20 Signal pin on digital 2
OneWire ds(DS18S20_Pin); // on digital pin 2

//Temperature chip i/o
void setup()
{
  Serial.begin(115200);
  Serial1.begin(115200);
  Serial.println("init...");
}

void loop()
{
  int sensorValue;
  int sensorValueA0 = analogRead(A0);// read the input on analog pin 0:
  float voltage = sensorValueA0* (5.0 / 1024.0); // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  Serial.println(voltage); // print out the value you read:
  delay(500);

  sensorValue = voltage;
  sensorValueA1 =analogRead(A1);
  sensorValueA2 = analogRead(A2);

  Serial1.print("(ID:1,A0:");
  Serial1.print(sensorValue);
  Serial1.print(",A1:");
  Serial1.print(sensorValueA1);
  Serial1.print(",A2:");
  Serial1.print(sensorValueA2);
  Serial1.print("}");
  Serial1.println("");
}

```

Figure 4 : Code 1 Antenne-Relais

```

delay(3000);

float temperature = getTemp();
Serial.println(temperature);

delay(100); //just here to slow down the output so it is easier to read
}

float getTemp(){
//returns the temperature from one DS18S20 in DEG Celsius

byte data[12];
byte addr[8];

if ( !ds.search(addr)) {
    //no more sensors on chain, reset search
    ds.reset_search();
    return -1000;
}

if ( OneWire::crc8( addr, 7) != addr[7]) {
    Serial.println("CRC is not valid!");
    return -1000;
}

if ( addr[0] != 0x10 && addr[0] != 0x28) {
    Serial.print("Device is not recognized");
    return -1000;
}

ds.reset();
ds.select(addr);
ds.write(0x44,1); // start conversion, with parasite power on at the end

byte present = ds.reset();

```

Figure 5 : Code 2 Antenne-Relais

```

float getTemp()
{
    //returns the temperature from one DS18S20 in DEG Celsius
    byte data[12];
    byte addr[8];
    if ( !ds.search(addr) ) {
        //no more sensors on chain, reset search
        ds.reset_search();
        return -1000;
    }
    if ( OneWire::crc8( addr, 7 ) != addr[7] ) {
        Serial.println("CRC is not valid!");
        return -1000;
    }
    if ( addr[0] != 0x10 && addr[0] != 0x28 ) {
        Serial.print("Device is not recognized");
        return -1000;
    }

    ds.reset();
    ds.select(addr);
    ds.write(0x44,1); // start conversion, with parasite power on at the end

    byte present = ds.reset();
    ds.select(addr);
    ds.write(0xBE); // Read Scratchpad

    for (int i = 0; i < 9; i++) { // we need 9 bytes
        data[i] = ds.read();
    }
    ds.reset_search();

    byte MSB = data[1];
    byte LSB = data[0];
    float tempRead = ((MSB << 8) | LSB); //using two's compliment
    float TemperatureSum = tempRead / 16;
    return TemperatureSum;
}

```

Figure 6 : Code 3 Antenne-Relais



Figure 7 : Test de flottabilité de la boîte (1)



Figure 8 : Test de flottabilité de la boîte (2)

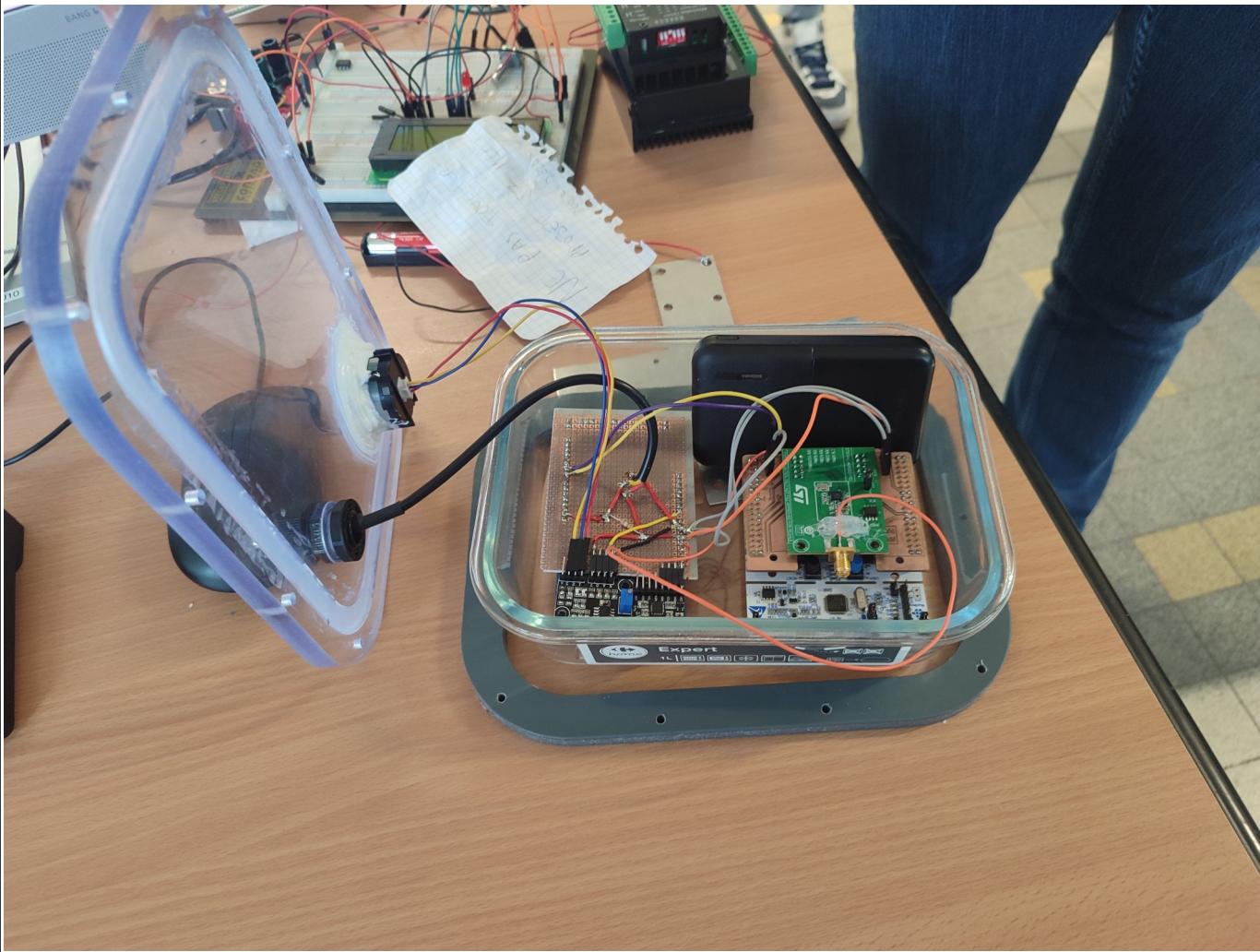


Figure 9 : Assemblage des différents capteurs dans la boîte

## ➤ **Bibliographie / Webographie**

[1] : *Biologie du vin et de la bière.* (s. d.). Planet-Vie.

<https://planet-vie.ens.fr/thematiques/microbiologie/biologie-du-vin-et-de-la-biere>

[2] : T, A. (2023, 1 novembre). *Foire aux questions fermentation.* Comment brasser sa bière.  
<https://comment-brasser-sa-biere.fr/faq-fermentation/>

[3] : Codezarduino. (2018, 18 avril). *Mesurer la qualité de l'eau : 1. La turbidité.* Codez Arduino pour la planète ! <https://codezarduino.wordpress.com/2018/02/02/mesurer-la-qualite-de-leau-1-la-turbidite/>

[4] : Turbide2\_web. (2018, 2 février). Codez Arduino pour la planète !

[https://codezarduino.wordpress.com/2018/02/02/mesurer-la-qualite-de-leau-1-la-turbidite/turbide2\\_web/](https://codezarduino.wordpress.com/2018/02/02/mesurer-la-qualite-de-leau-1-la-turbidite/turbide2_web/)

[5] : Tronic, G. (s. d.). *Capteur de Turbidit Grove 101020752.* GO TRONIC.

<https://www.gotronic.fr/art-capteur-de-turbidite-grove-101020752-31769.htm>

[6] : Tronic, G. (s. d.-b). *Capteur de turbidit SEN0189.* GO TRONIC. <https://www.gotronic.fr/art-capteur-de-turbidite-sen0189-27864.htm>

[7] : Robots, D. F. (2020). *Waterproof\_DS18B20\_Digital\_Temperature\_Sensor\_SKU\_DFR0198\_-DFRobot.* wiki. dfrobot. com. [https://wiki.dfrobot.com/Waterproof\\_DS18B20\\_Digital\\_Temperature\\_Sensor\\_SKU\\_DFR0198\\_](https://wiki.dfrobot.com/Waterproof_DS18B20_Digital_Temperature_Sensor_SKU_DFR0198_) (Accessed Dec. 11, 2023).

[8] : Tronic, G. (s. d.-d). *Sonde PH + Interface SEN0161V2.* GO TRONIC. <https://www.gotronic.fr/art-sonde-ph-interface-sen0161v2-28709.htm>

[9] : Tronic, G. (s. d.-b). *Sonde PH + Interface Pro SEN0169.* GO TRONIC.

<https://www.gotronic.fr/art-sonde-ph-interface-pro-sen0169-24570.htm>