INSEA

المملكة المغربية

وزارة التربية الوطنية والتعليم الأولي والرياضة

TRAVAUX D'INITIATIVE PERSONNELLE

ENCADRÉS T.I.P.E. 2026

CYCLE , BOUCLE

SUJET :

Gestion intelligente et automatisée de la qualité de l'eau d'un aquarium scientifique

Encadré par :

- M Y. RAHOU
- M A. OUAANABI

Préparé par :

- TAHRI SAFOUANE

PLAN

Introduction

Pour étudier une espèce biologique en laboratoire, il est essentiel de lui offrir un milieu qui ressemble le plus possible à son habitat naturel. Dans le cas des organismes aquatiques, le contrôle régulier des paramètres de l'eau est indispensable pour maintenir l'équilibre biologique du milieu, En effet, le moindre déséquilibre de l'eau peut conduire à leur mortalité et fausser les observations scientifiques.

STATISTIQUES SUR LA DURÉE DE VIE DES POISSONS DANS LES AQUARIUMS ORDINAIRES

❑ **les poissons ont souvent une espérance de vie réduite dans les aquariums ordinaires**

10 à 50 %

Taux de mortalité annuel dans un aquarium domestique classique.

20 à 80 %

Taux de mortalité pouvant être observé la première année dans de mauvaises conditions.

Durée de vie réelle réduite

Souvent 2 à 4 fois plus faible que son potentiel biologique dans de mauvaises conditions.

Exemple – Poisson néon

Durée de vie potentielle : 5 à 8 ans

Durée de vie en aquarium mal entretenu : souvent moins de 2 ans.

Pour cela, ce sujet vise à concevoir **un milieu convenable aux poissons**. Le système doit surveiller les paramètres essentiels qui les affectent directement, afin d'assurer les conditions de vie permettant de prolonger leur durée de vie.

TEMPÉRATURE

TURBIDITÉ

❑ Mesurer et réguler ces grandeurs

les sources : EnviroLiteracy – What is the death rate of aquarium fish? | IERE – What is the mortality rate for fish in aquariums? | Gensou – Aquarium Fish Lifespan Chart Guide

Problématique :

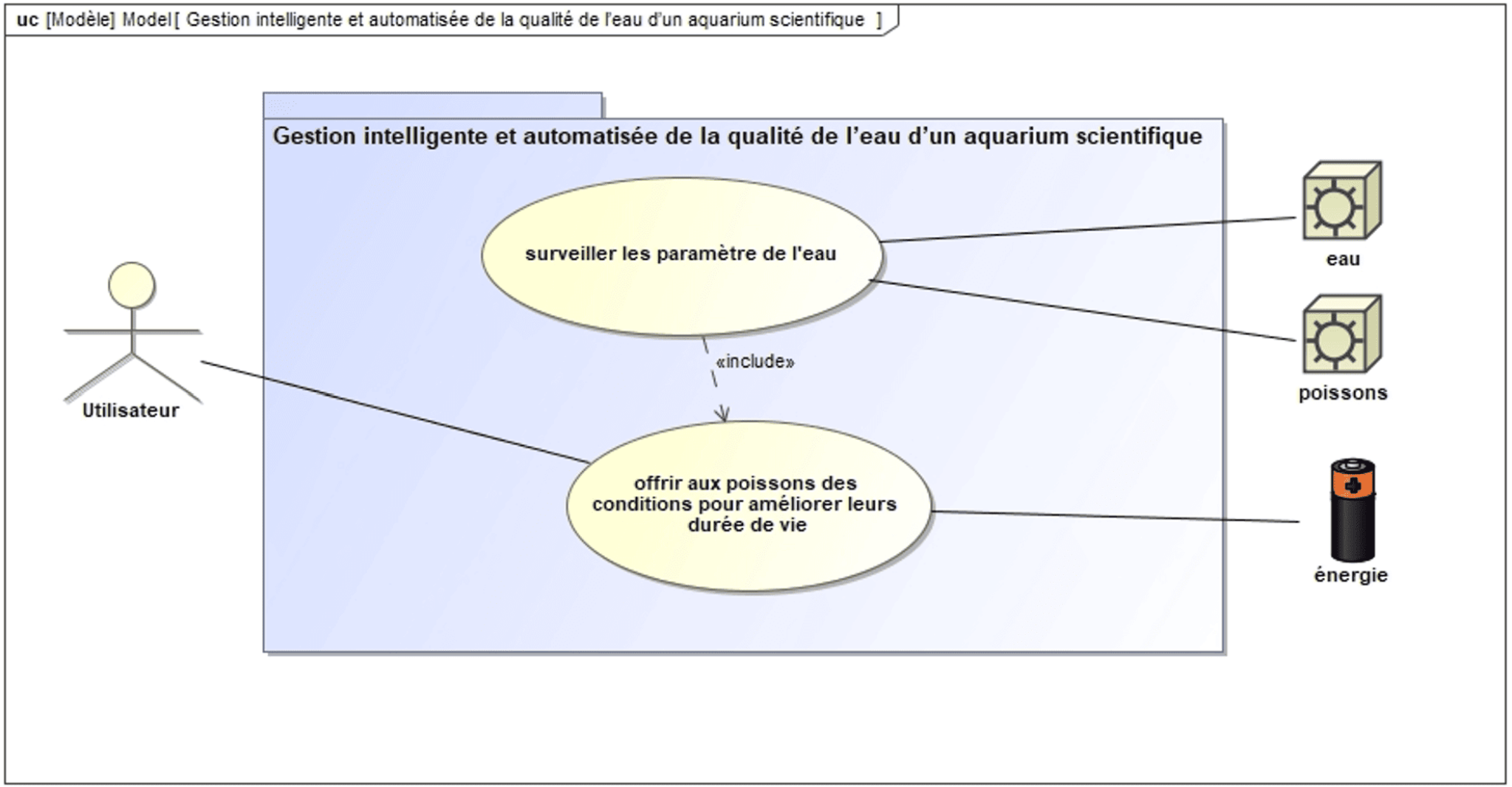
La qualité de l'eau constitue un élément essentiel à la survie des poissons en aquarium. Dès lors, comment concevoir un système intelligent capable de surveiller en continu les paramètres clés de l'eau et d'assurer leur régulation automatique afin de maintenir un environnement stable ?

❑Objectifs

- Mesurer et réguler la température de l’eau.
- Surveiller la turbidité et filtrer afin d’assurer un eau claire.
- Dimensionner et choisir le matériau de l'aquarium.
- Réaliser une interface de communication avec l'utilisateur
- Réaliser un prototype de l'aquarium

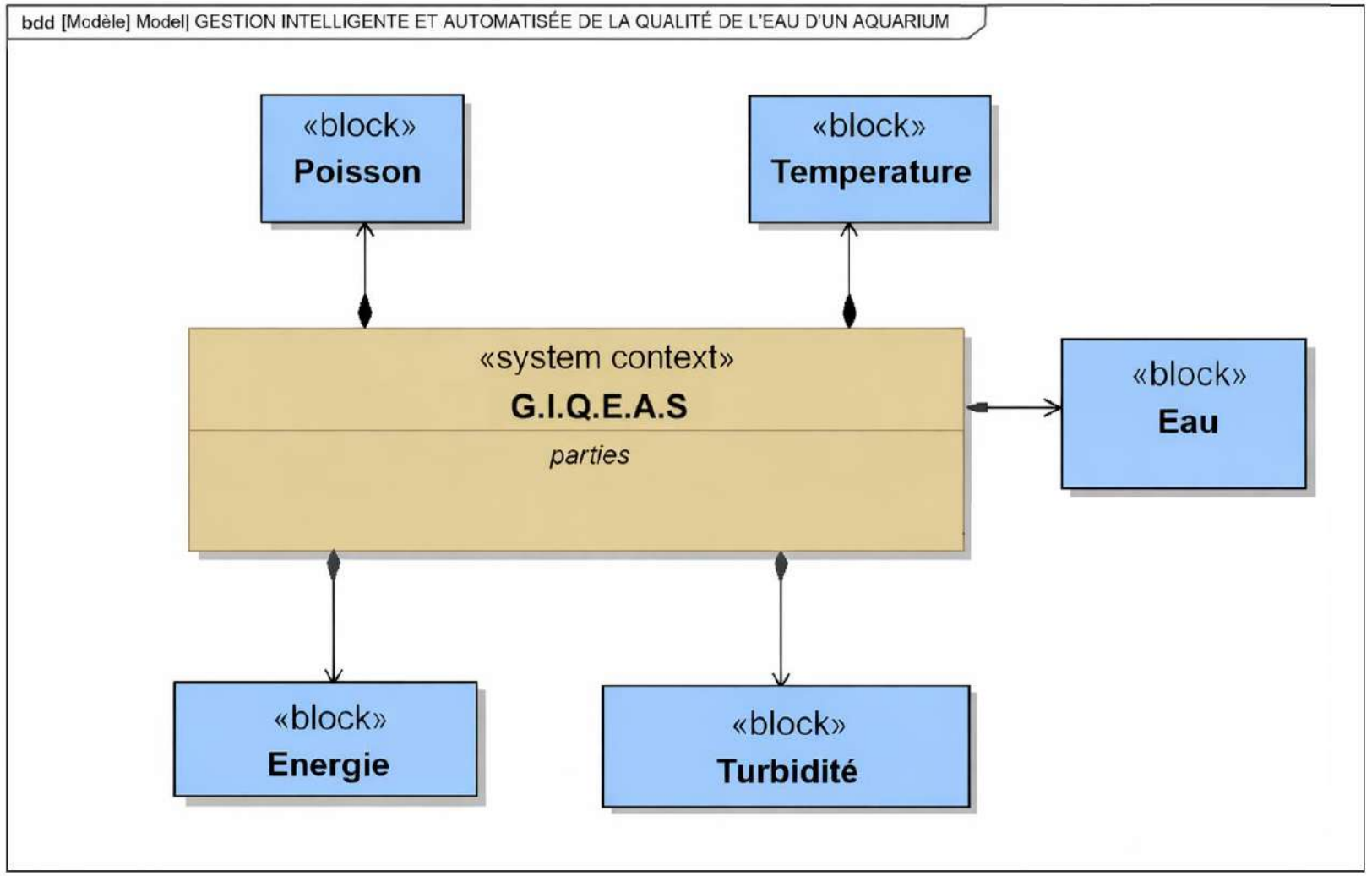
Fonctionnement du système

❑ Diagramme de cas d'utilisation



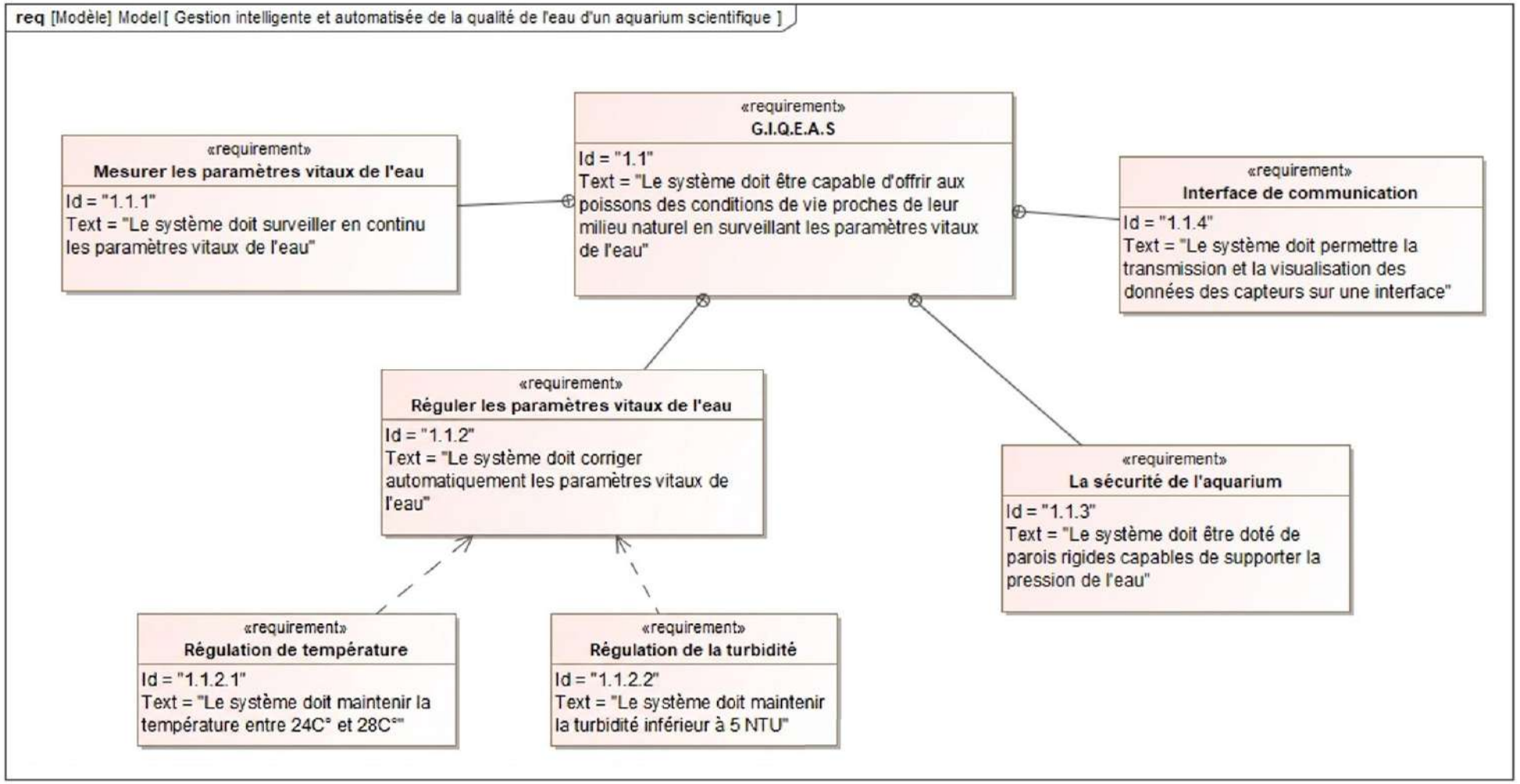
Fonctionnement du système

❑ Diagramme BDD contexte



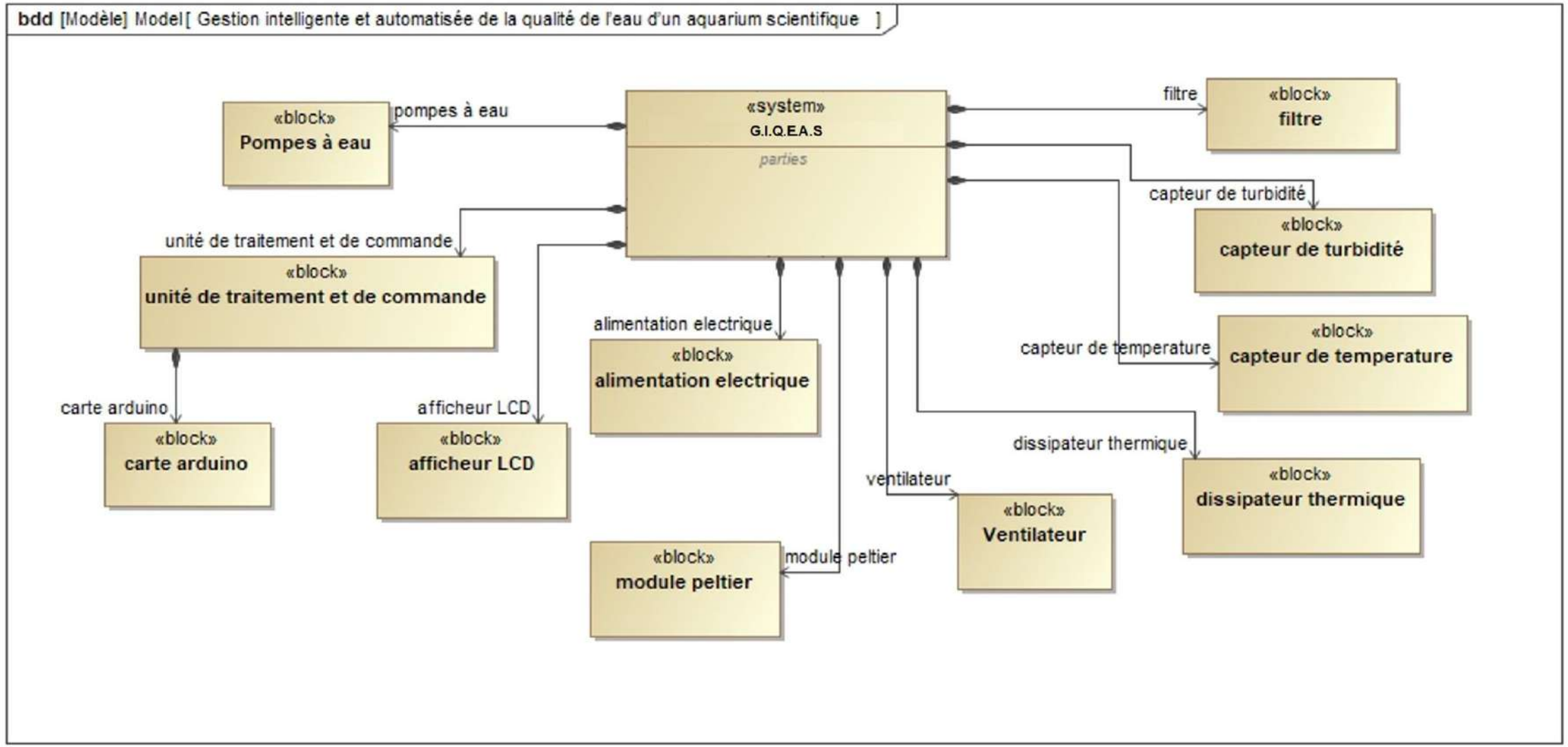
Fonctionnement du système

❑ Diagramme d'exigence

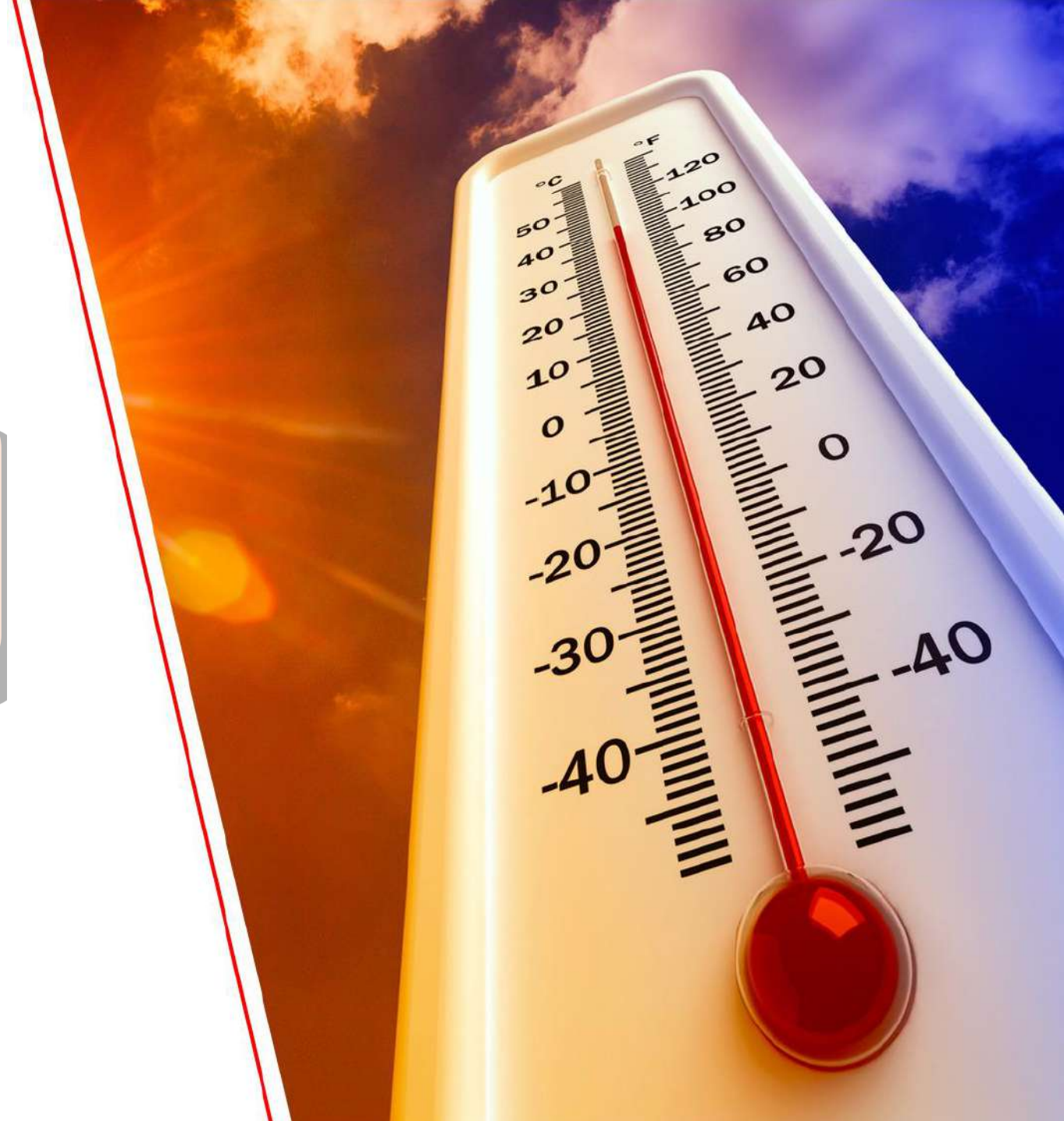


Fonctionnement du système

❑ Diagramme BDD

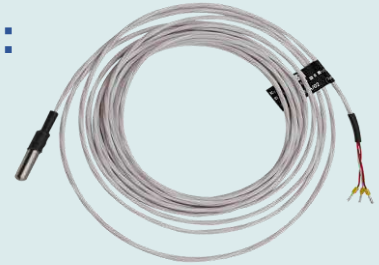


objectif 1 :
Réguler la température de l’eau



Analyse des solutions : choix du capteur :

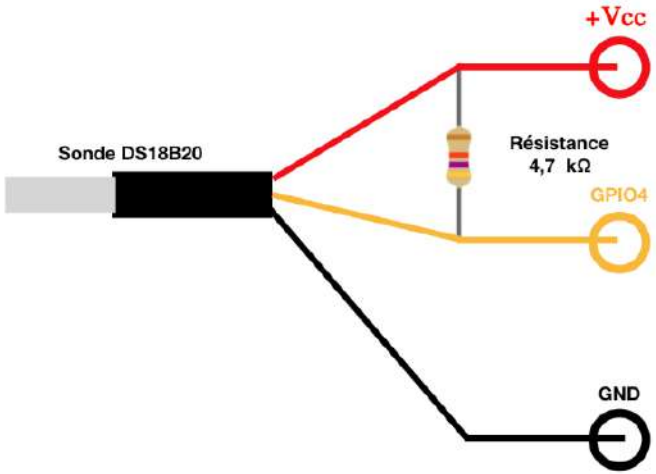
❑ Comparaison : PT100, DS18B20 et NTC

Caractéristiques	PT100: 	DS18B20: 	NTC: 
Plage de mesure	−200 °C à +850 °C	−55 °C à +125 °C	−40 °C à +150 °C
Précision	±0,43 °C	±0.5 °C	±1 °C à ±3 °C
Avantages	✓ Grande stabilité ✓ Très précis et linéaire ✓ Large plage de mesure	✓ Très facile à utiliser ✓ Sortie numérique (pas de conversion/filtrage externe)	✓ Sensible à de petits changements de T ✓ Facile à intégrer
Inconvénients	✗ Coût plus élevé ✗ Nécessite une interface (pont de Wheatstone)	✗ Plage de température limitée (mais suffisante)	✗ Caractéristique non linéaire → nécessite calcul ✗ Précision moyenne

Conclusion : Le capteur DS18B20 est le meilleur choix, car il offre une bonne précision, une grande facilité d'utilisation et un coût inférieur par rapport aux autres capteurs.

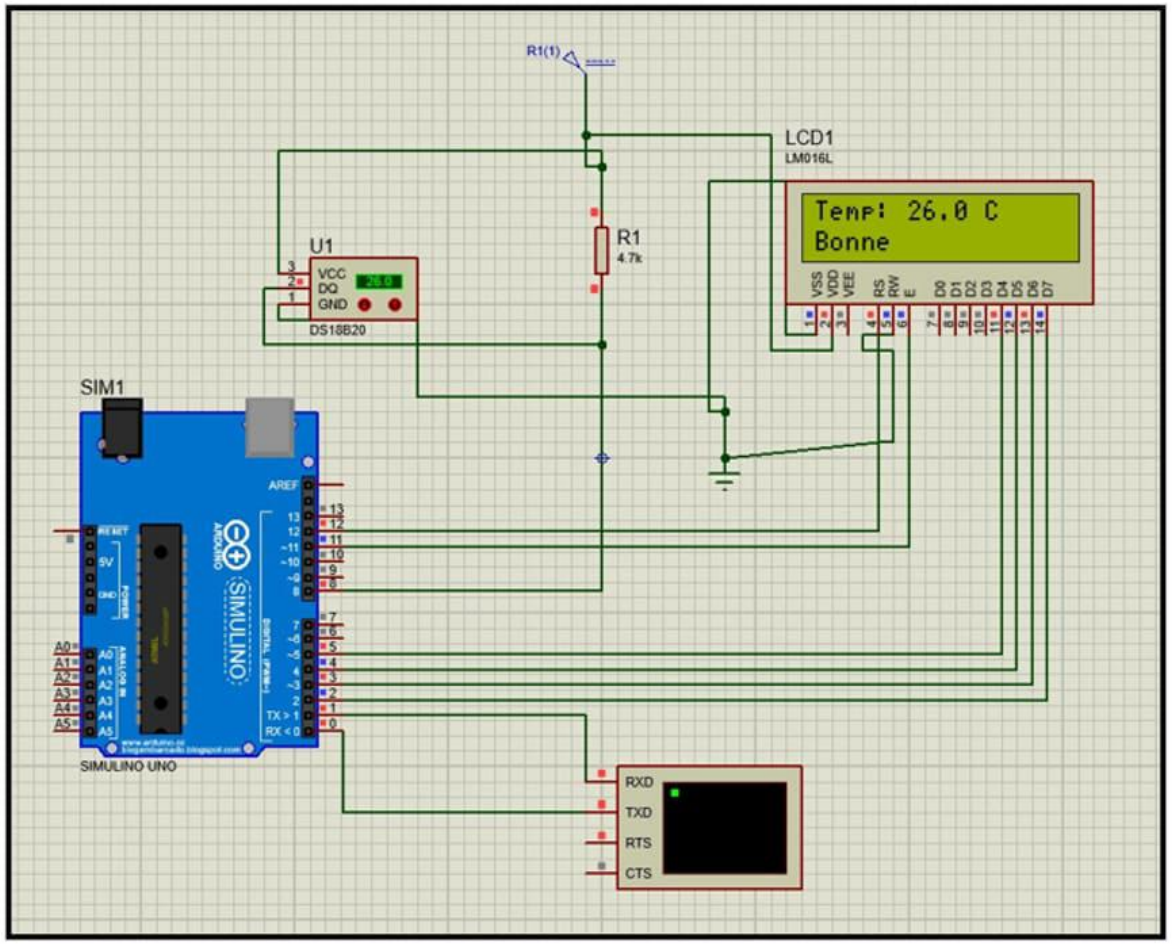
conditionnement de signal :

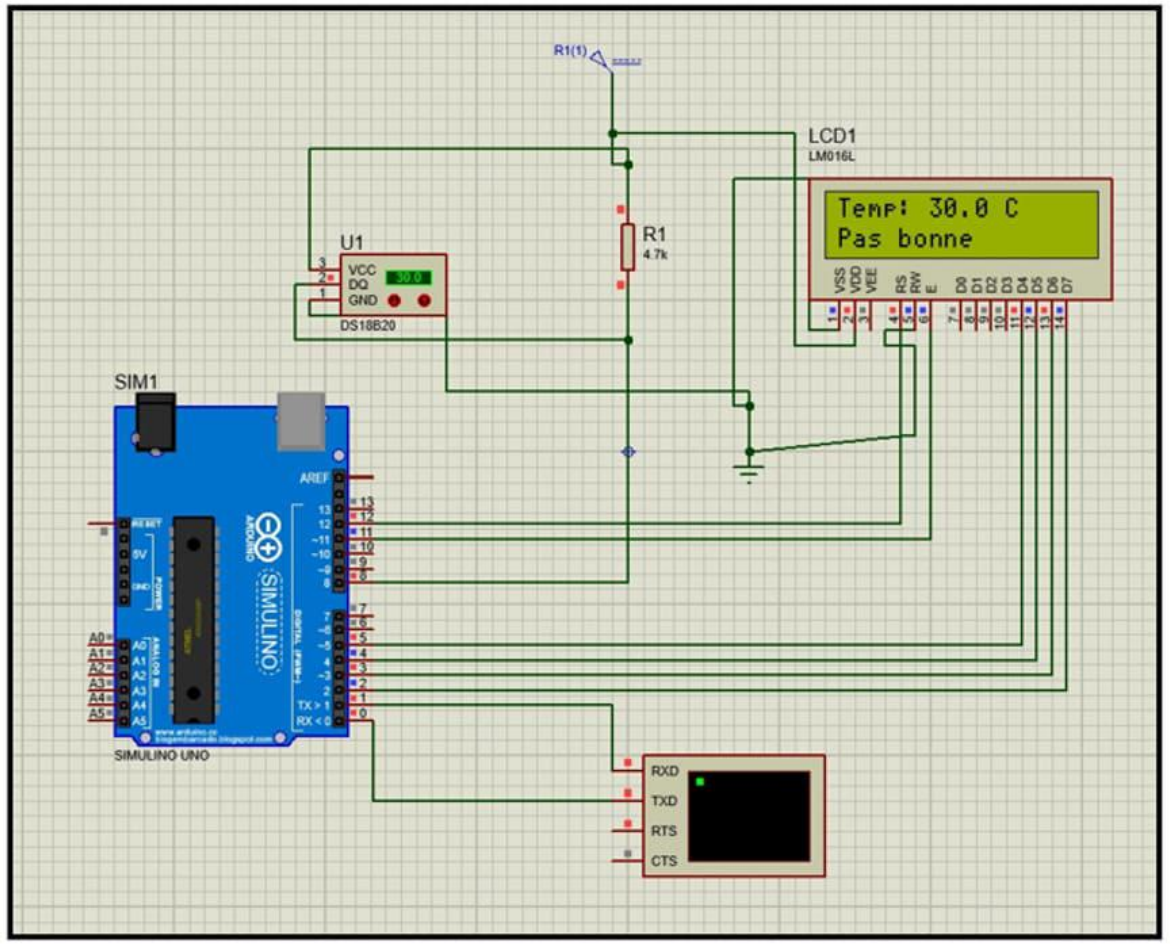
Le DS18B20 utilise le protocole **1-Wire** :
la même ligne sert à **envoyer** et **recevoir** des données.
Ni l'Arduino ni le capteur ne maintiennent la ligne au niveau haut de manière permanente.
Donc on met une **résistance pull-up entre DATA et +5V** pour que la ligne soit **toujours haute**



Proteus Isis

❑Test des cas critiques :



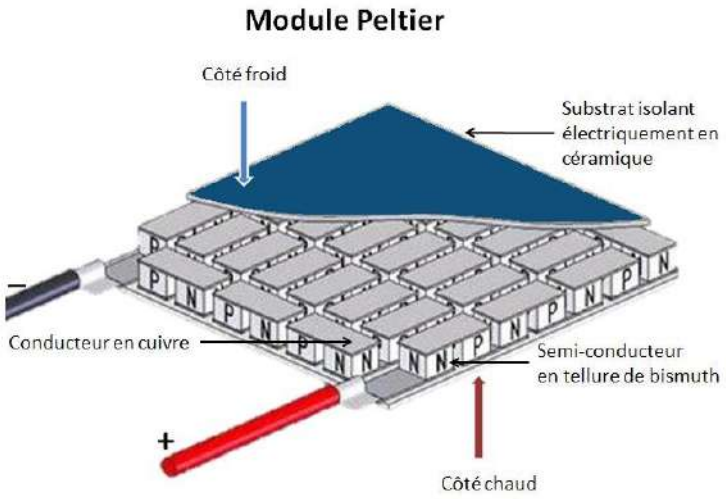


Refroidissement

La **TEC1-12706** est une plaque thermoélectrique basée sur l'effet Peltier, Lorsqu'un courant électrique la traverse, elle transfère la chaleur d'un côté vers l'autre : une face devient froide tandis que l'autre devient chaude. En inversant la polarité, les faces s'inversent également. Ce module est souvent utilisé pour le refroidissement ou le chauffage des systèmes

❑ La formule principale (1er principe) :

$$Qc = Qf + W$$



Refroidissement

❑ Caractéristiques du module :

Tension	12 V
Puissance max (Qfmax)	55 W
Température max	80 °C
Température min	–40 °C
Courant maximal	6 A

❑ Le COP — Coefficient de Performance :

$$COP = Qf / W$$

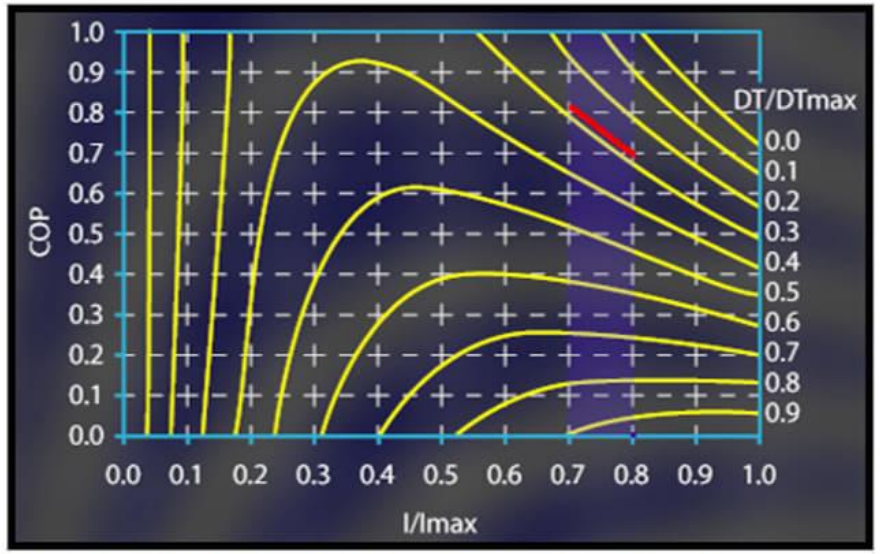
Les abaques techniques du TEC1-12706 indiquent **que** la valeur **Qf = 14 w** correspond à sa puissance frigorifique utile réelle, extraite selon les courbes de la datasheet.

Source : https://www.thermonamic.com/TEC1-12715-English%2020220526.pdf

Or $W = V \times I \approx 60\ w$

Donc : $COP = 0{,}23$

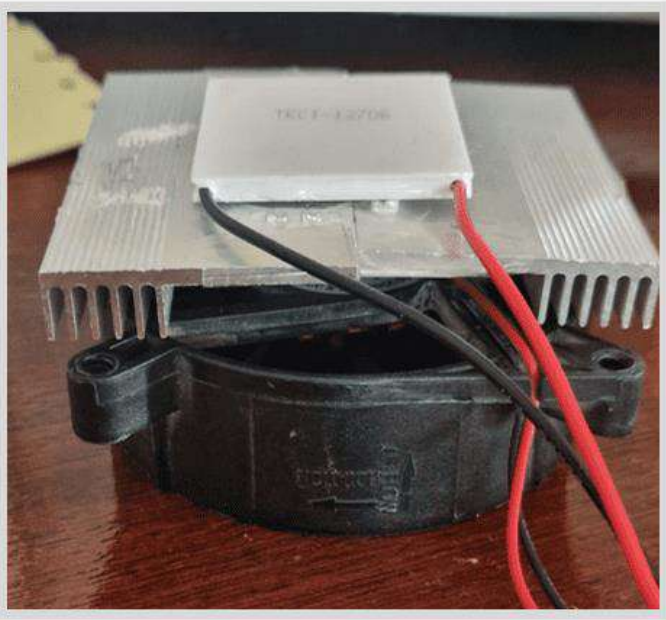
Le module fonctionne ici dans une zone de faible rendement



Le meilleur COP est généralement compris entre 0.3 et0.7

Refroidissement

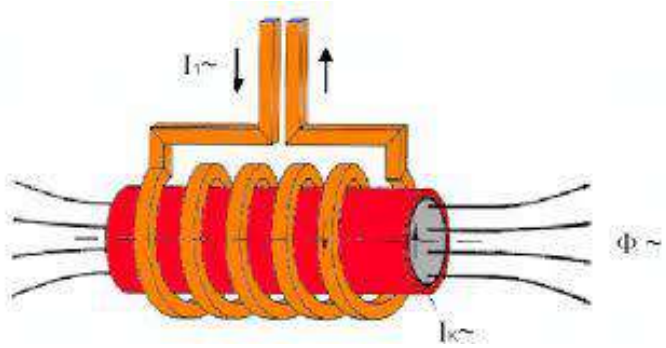
- Pour améliorer l'efficacité, j'ai installé un radiateur plus grand que la plaque et **un** ventilateur plus puissant sur la face chaude du module.

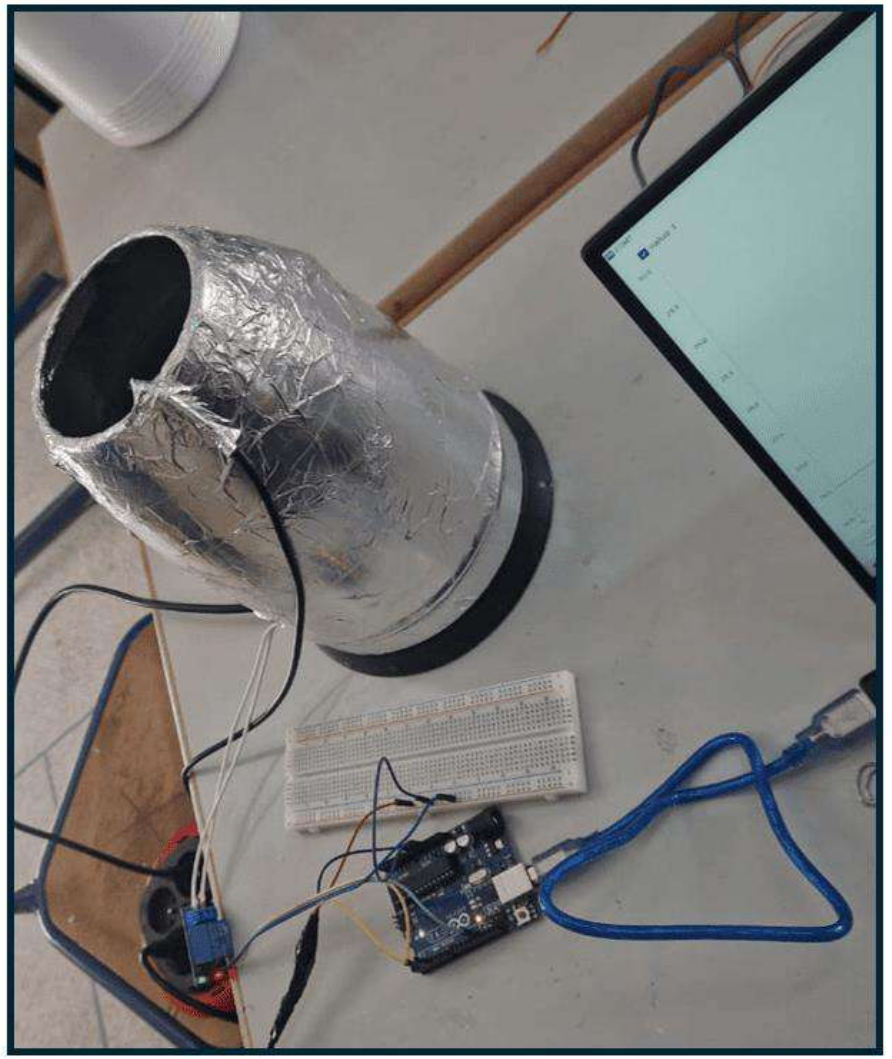


Cela diminue ΔT ce qui augmente la puissance frigorifique Q_f, et par conséquent, augmente directement le COP du système.

Réchauffement

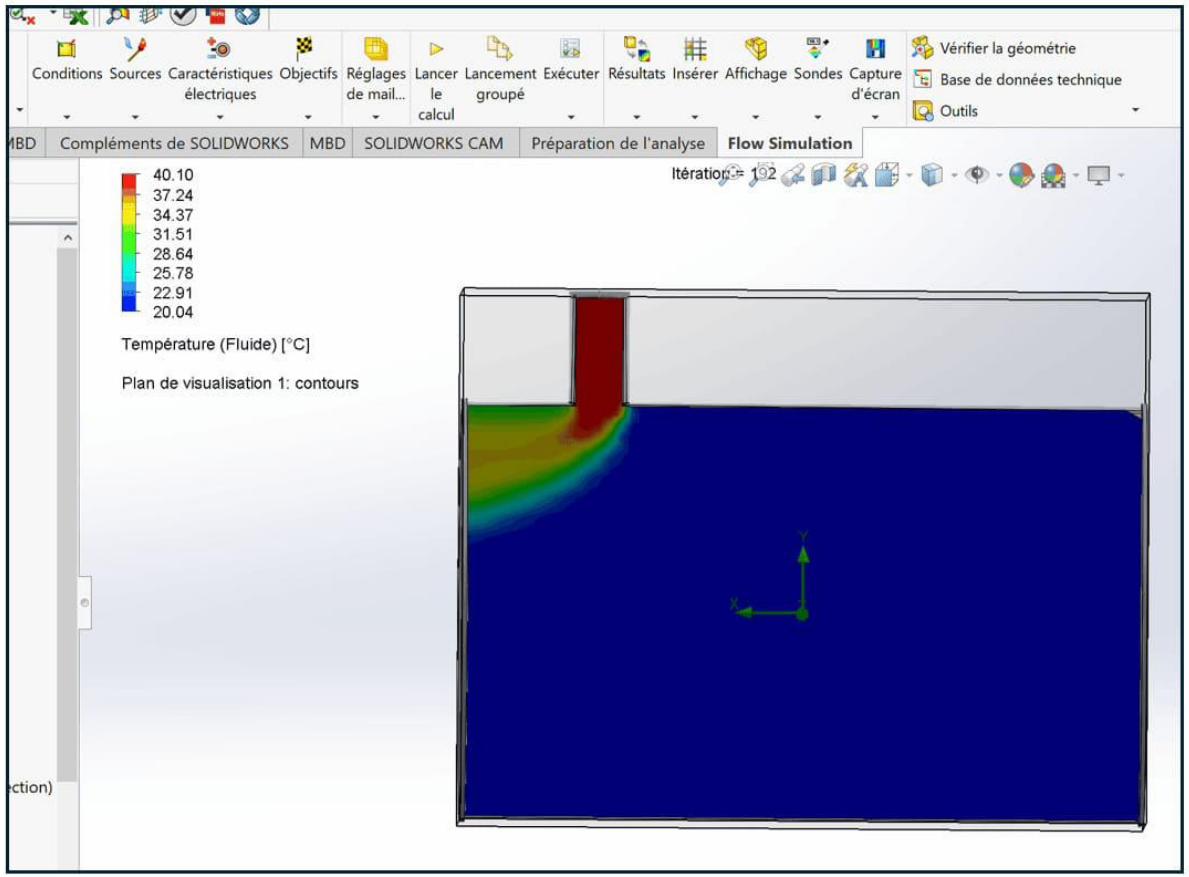
Un **réservoir** est équipé d'une **résistance chauffante** interne destinée à chauffer l'eau. Le système est contrôlé par un relais qui coupe automatiquement l'alimentation électrique lorsque la température dépasse un seuil prédéfini, afin d'éviter toute surchauffe.

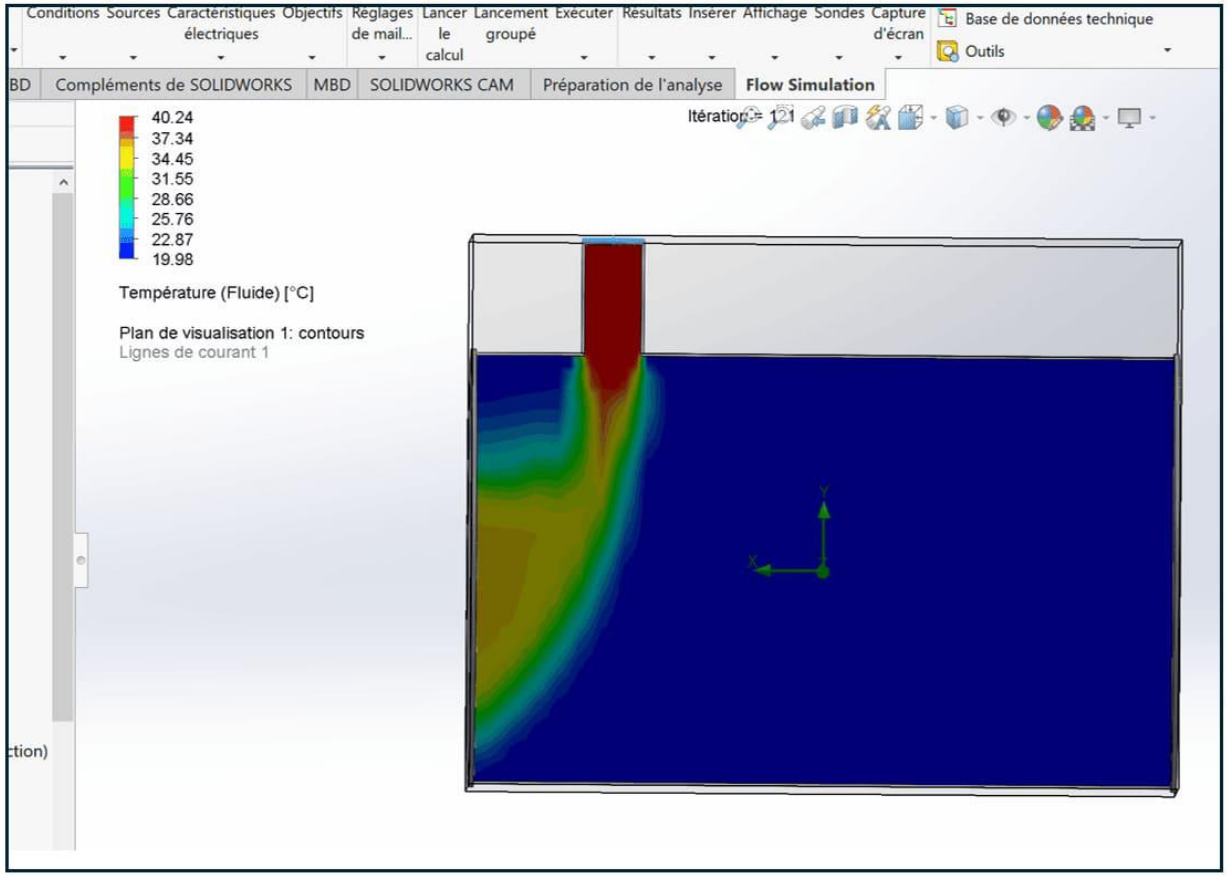


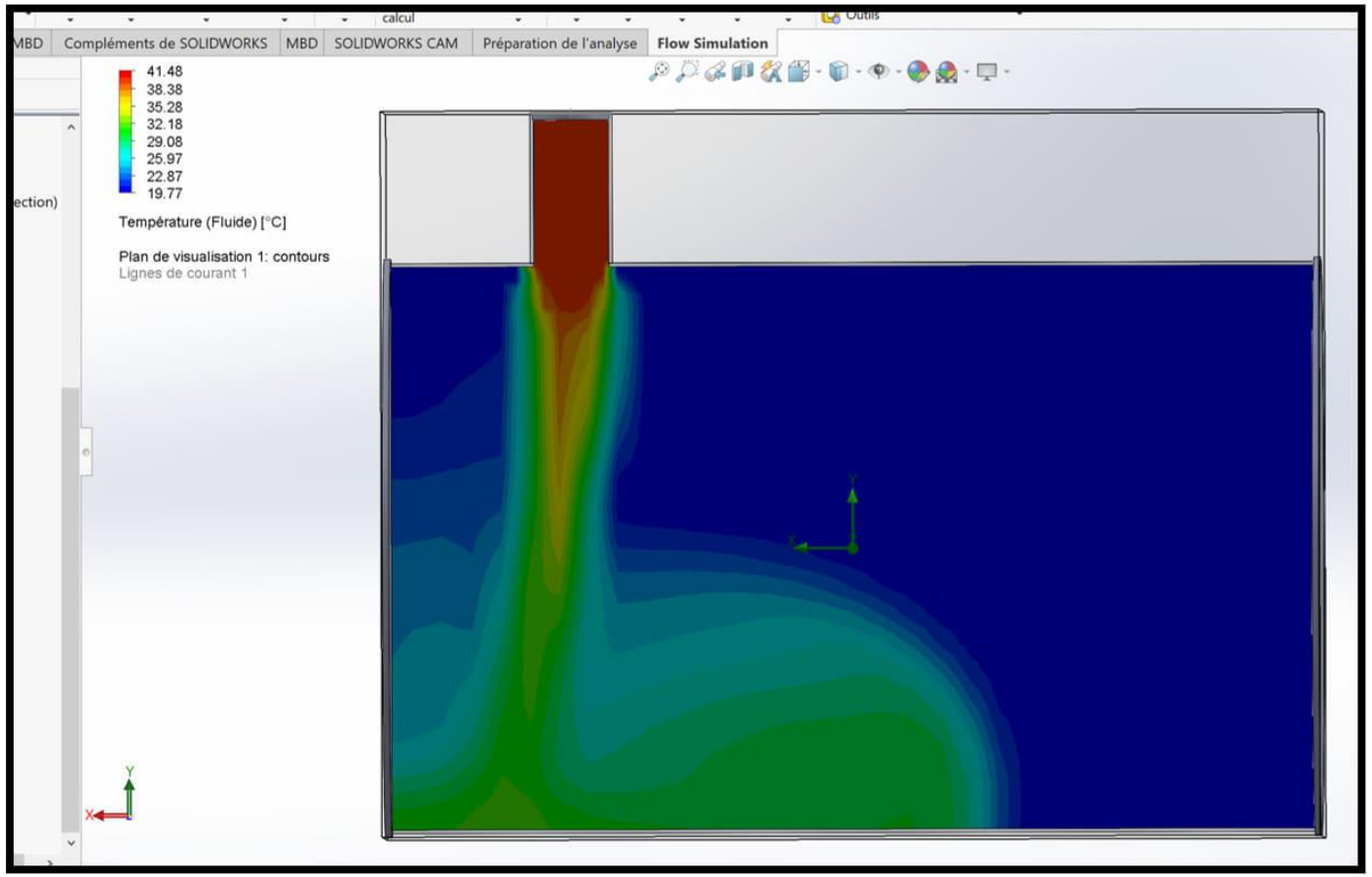


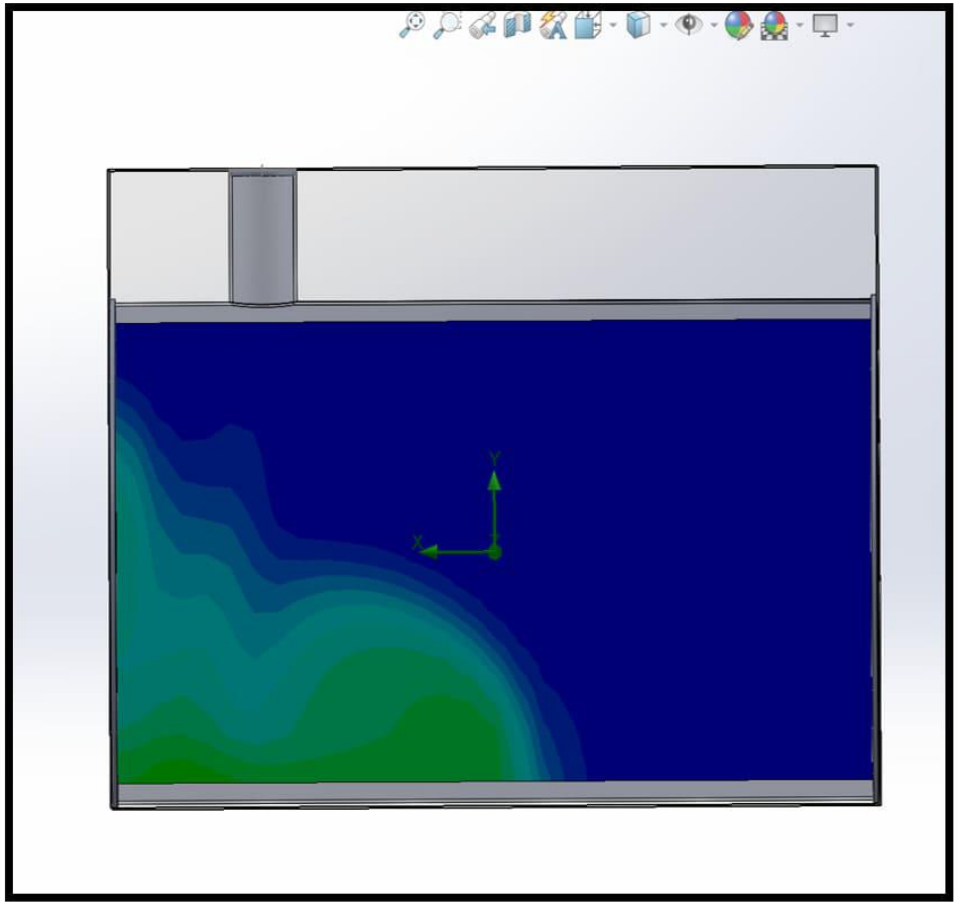
Représentation du mélange thermique de l'eau chaude et de l'eau froide

Injection de l'eau chaude dans l'eau froide







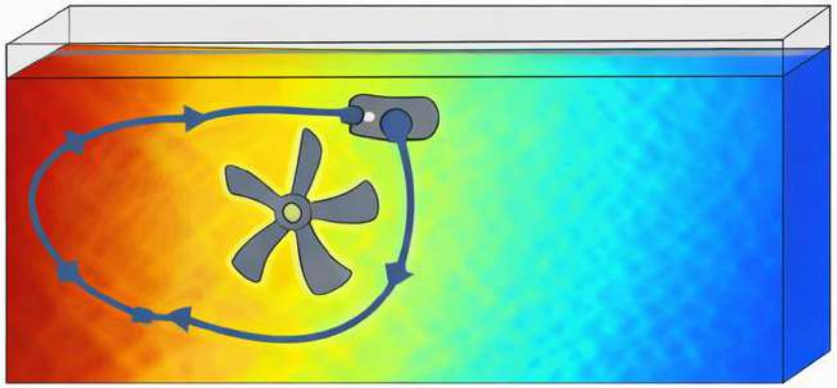


Le résultat indique un problème : la zone proche de la source d'écoulement chaude présente une température plus élevée par rapport aux zones éloignées.

Solutions proposées pour résoudre ce problème

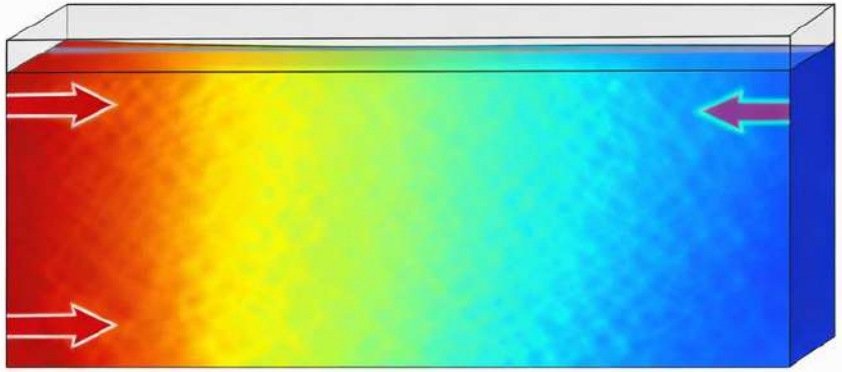
1.Installer une petite turbine fonctionnant à faible vitesse de rotation afin d'améliorer le mélange de l'eau.

Solution 1



Solution 2

2.Ajouter plusieurs points d'injection d'eau chaude répartis stratégiquement afin d'optimiser la répartition de la température.



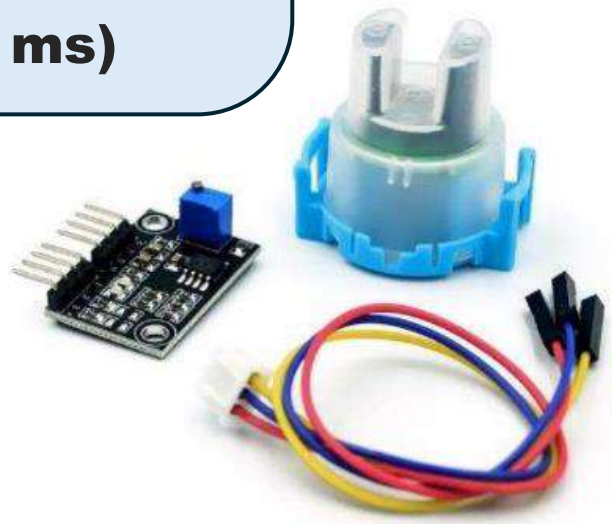
Objectif 2 :
Filtrer l’eau

Analyse des solutions : choix du capteur

❑ Régulation de la turbidité : Capteur de la Turbidité TS–300B

✓ **Sortie analogique + numérique (seuil réglable) compatible avec Arduino sans circuit compliqué**

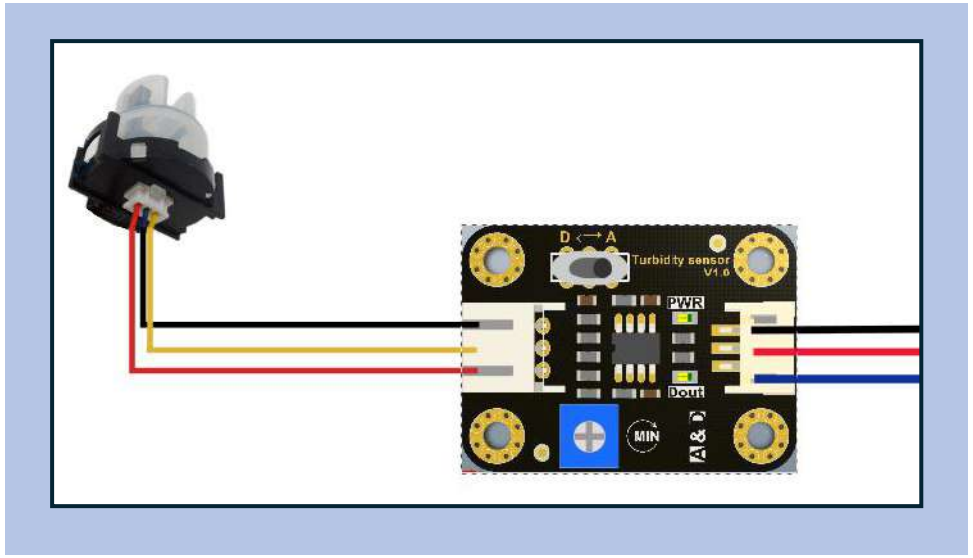
✓ **Rapide (réponse < 500 ms)**

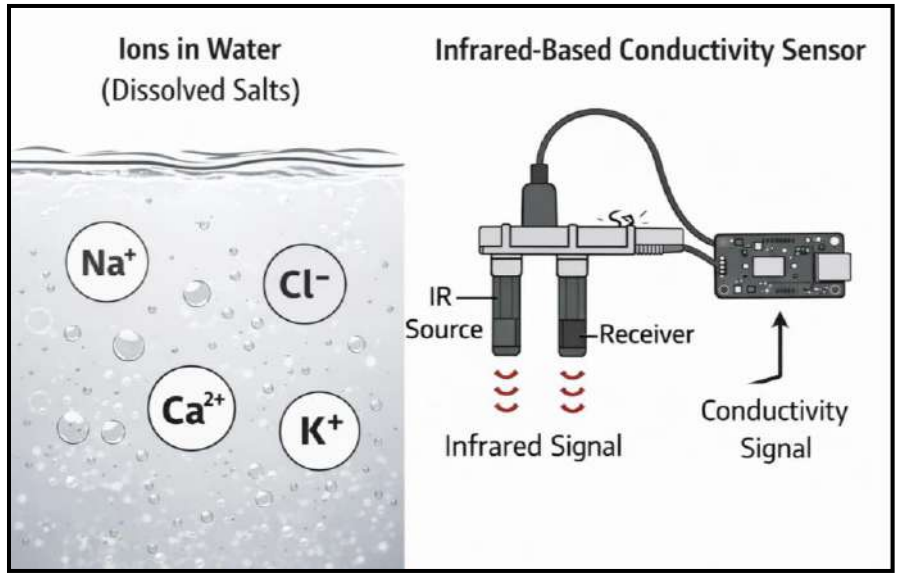


- Operating temperature: -20 ° C ~ 90 ° C
- Module size: 38.6mm*22.1mm
- Sensor Interface: XH2.54 connector
- **Ratio Range (NTU): 0~1000±30**
- Infrared Emitting Diode: 940nm (Peak emission wavelength)
- Photo Transistor: 880nm (Peak emission wavelength)
- Operating voltage: 5.00V DC
- Working current: 40mA (MAX)
- Response time: <500ms
- Insulation resistance: 100MΩ (Min)
- **Output mode: analog output: 0~4.5V; digital output: high/low level signal (can adjust the potentiometer to select the corresponding threshold)**

Principe de fonctionnement

Le **capteur TS-300B** fonctionne grâce à un système optique composé d'une LED infrarouge et d'un phototransistor. La LED émet de la lumière à travers l'eau, puis le phototransistor mesure la quantité de lumière reçue. Lorsque l'eau est claire, la lumière traverse facilement.





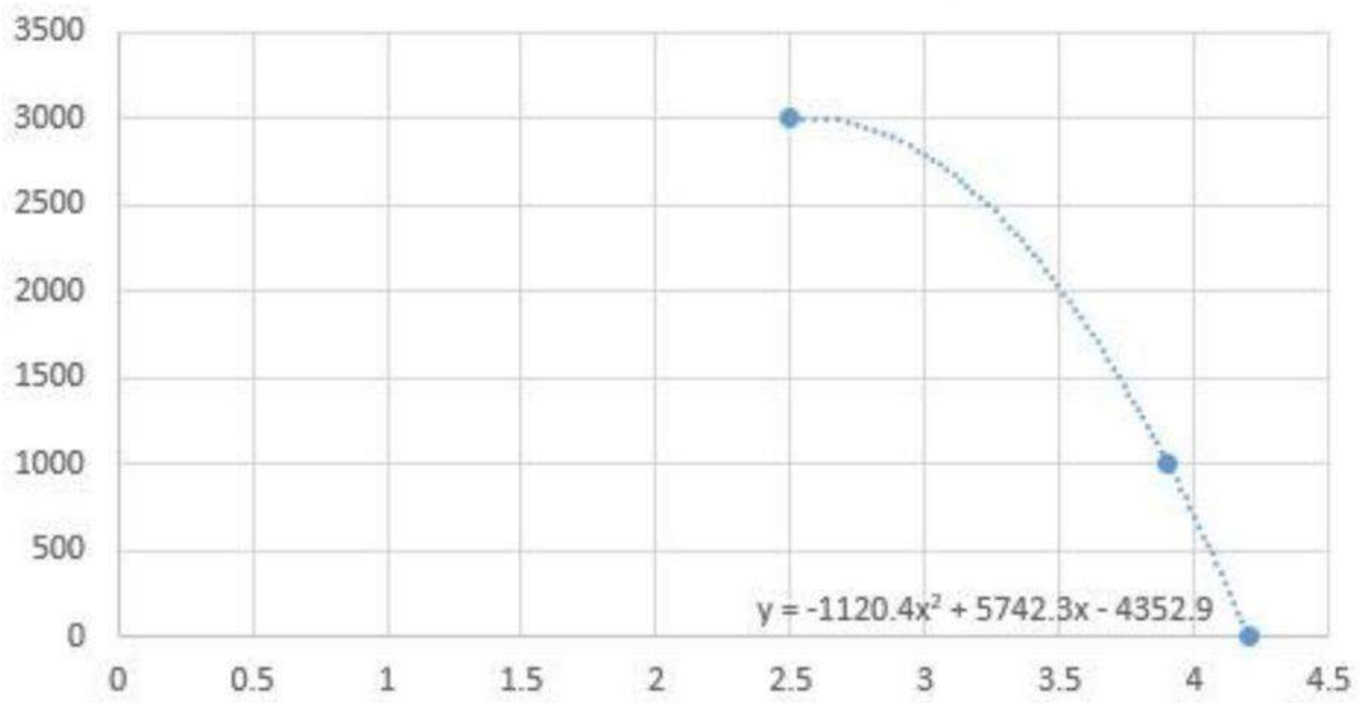
Relation : tension → turbidité

la relation expérimentale fournie par le fabricant est généralement :

$$\text{Turbidité (NTU)} = -1120.4 \times V^2 + 5742.3 \times V - 4352.9$$

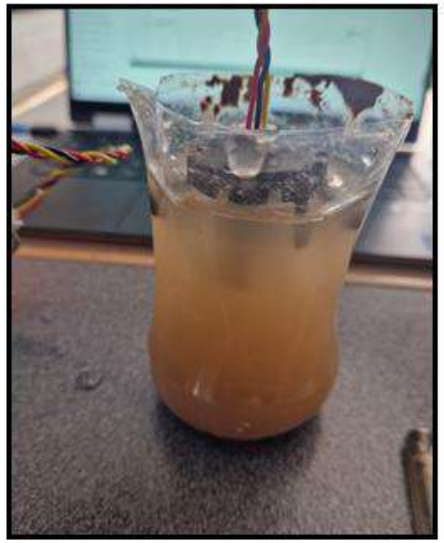
où : V= tension analogique en **volts** (0–4.5 V environ)

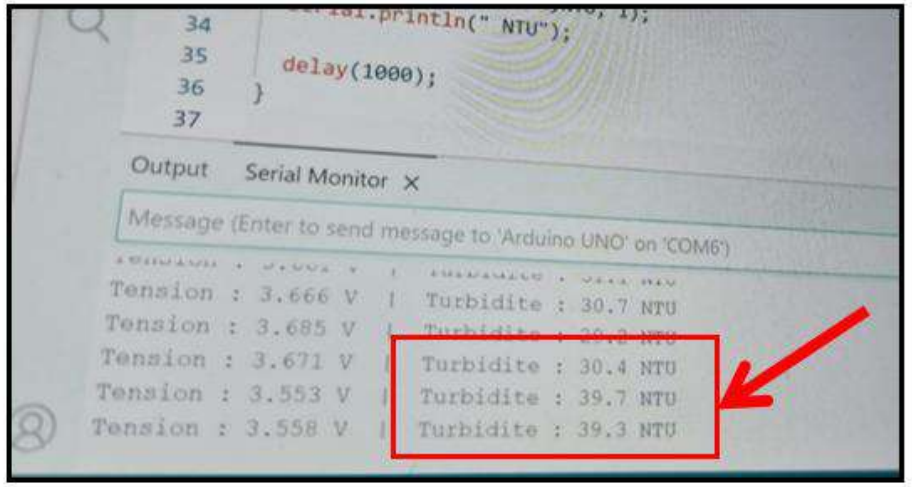
Tension V (en volts)

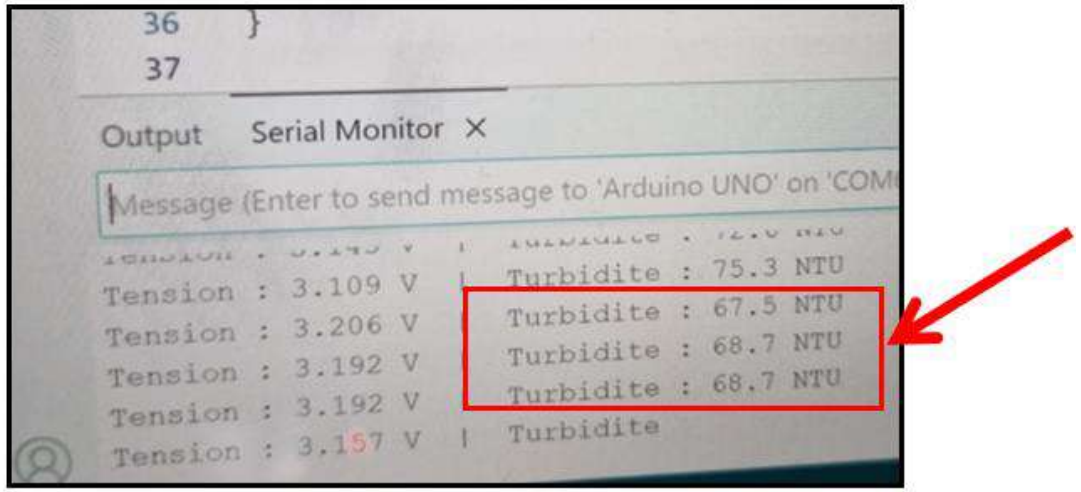


Turbidité (NTU)

Expérience pour mesurer la turbidité d'eau sale

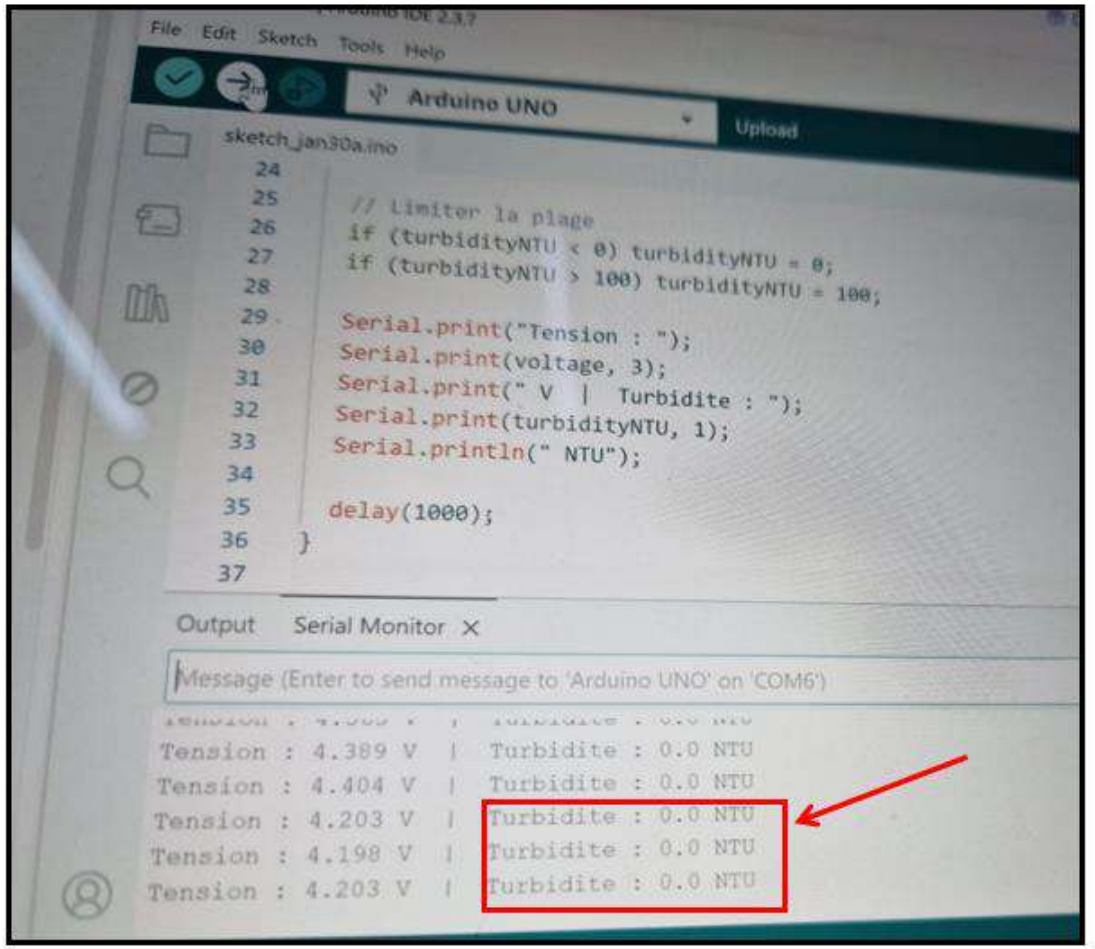






Expérience pour mesurer la turbidité d'eau propre

UNT <5 : eau claire
5 < UNT < 30 : eau légèrement trouble
UNT > 50 : eau trouble



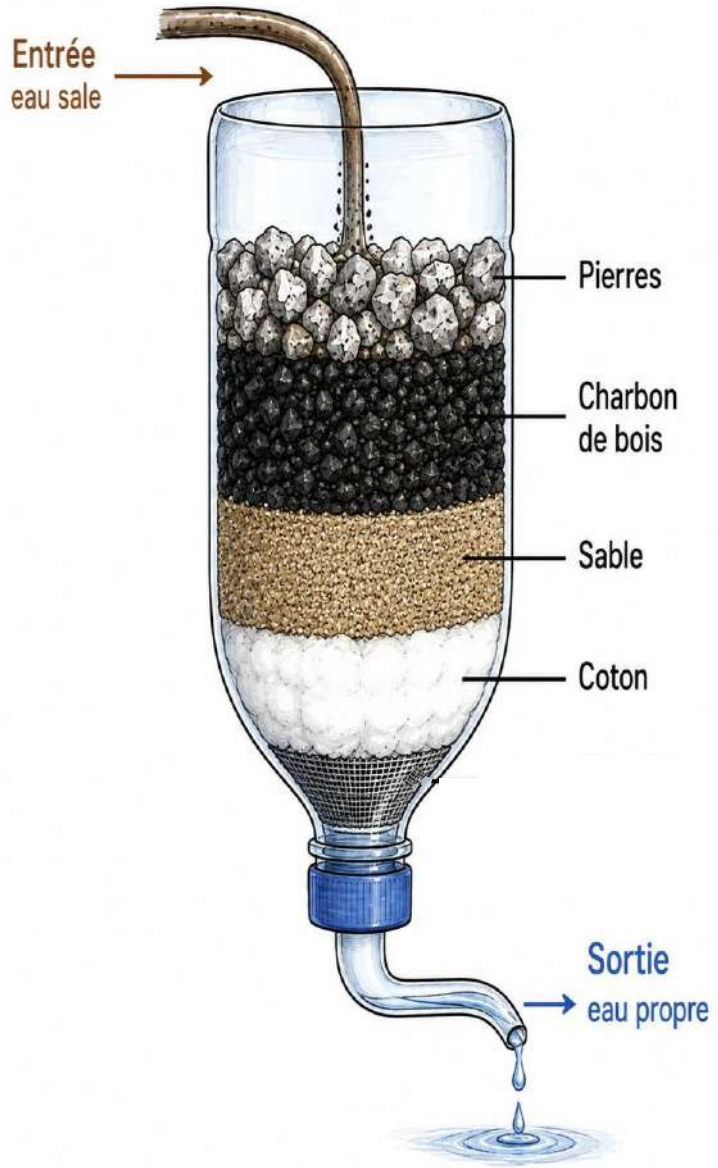
Conclusion : Le capteur de la turbidité est correctement calibré, selon les résultats de l'expérience.

Régulation de la Turbidité

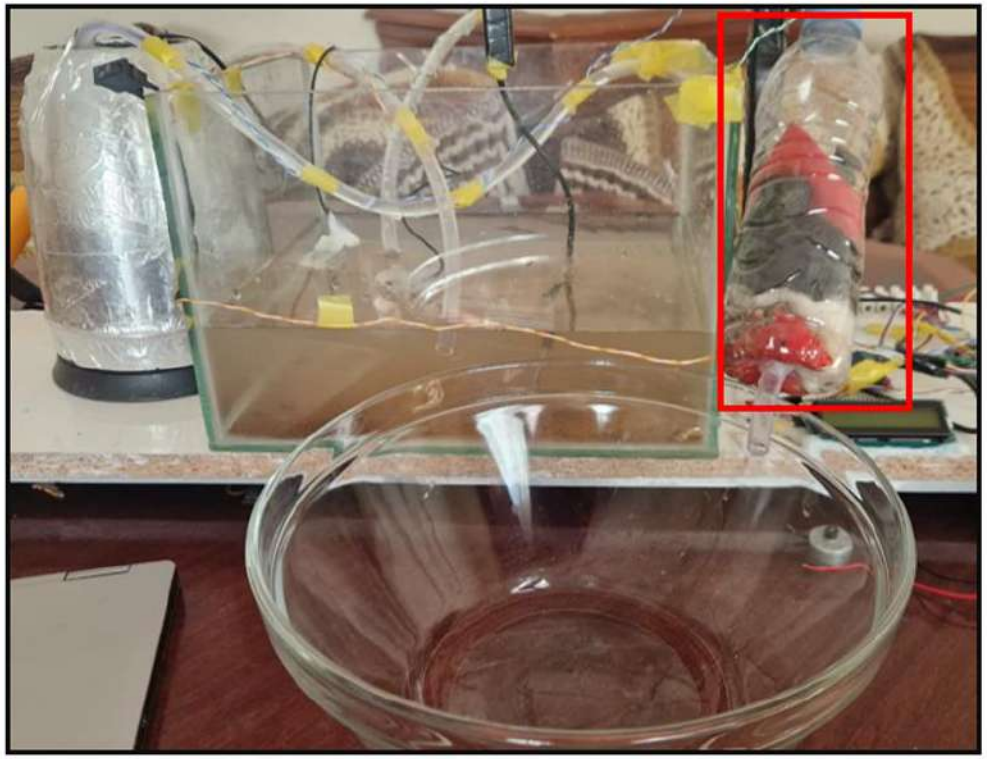
❑ Système de Filtration :

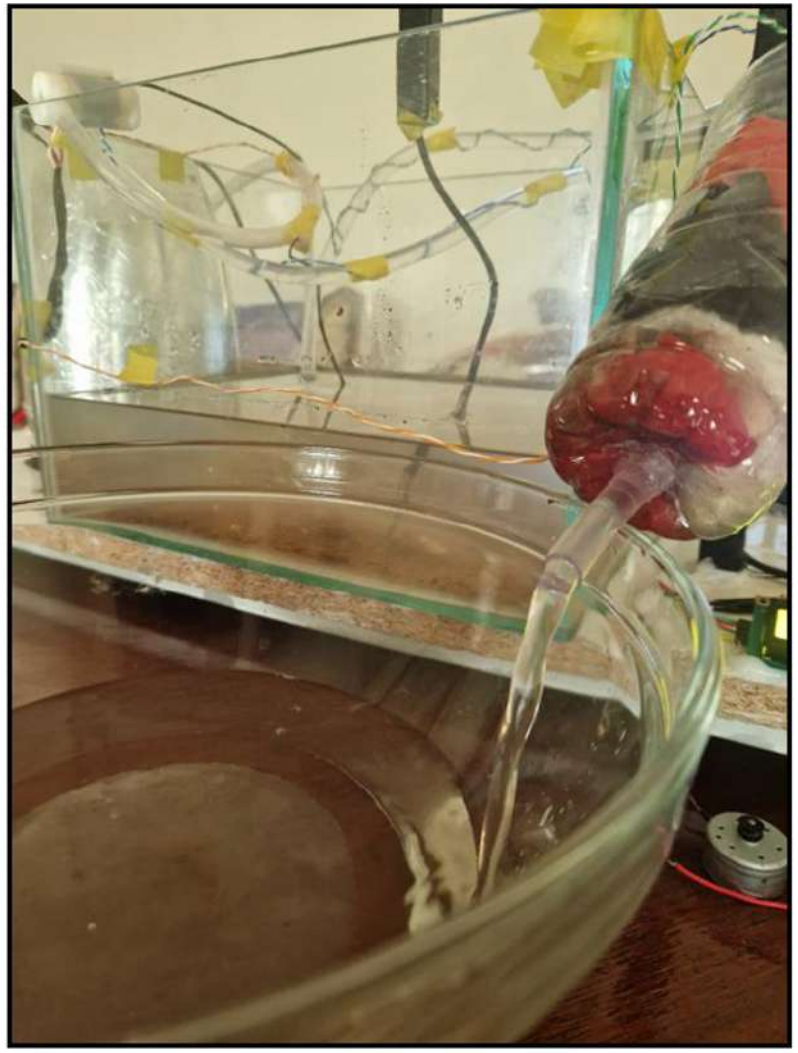
Ce filtre artisanal purifie l'eau en la faisant passer par quatre couches distinctes.

- Les pierres en haut captent les gros débris.
- Le charbon de bois adsorbe les impuretés chimiques et organiques, améliorant le goût et l'odeur.
- La couche de sable fin élimine les particules plus petites et la turbidité.
- Enfin, un tampon de coton en bas agit comme un filtre final pour les dernières impuretés.

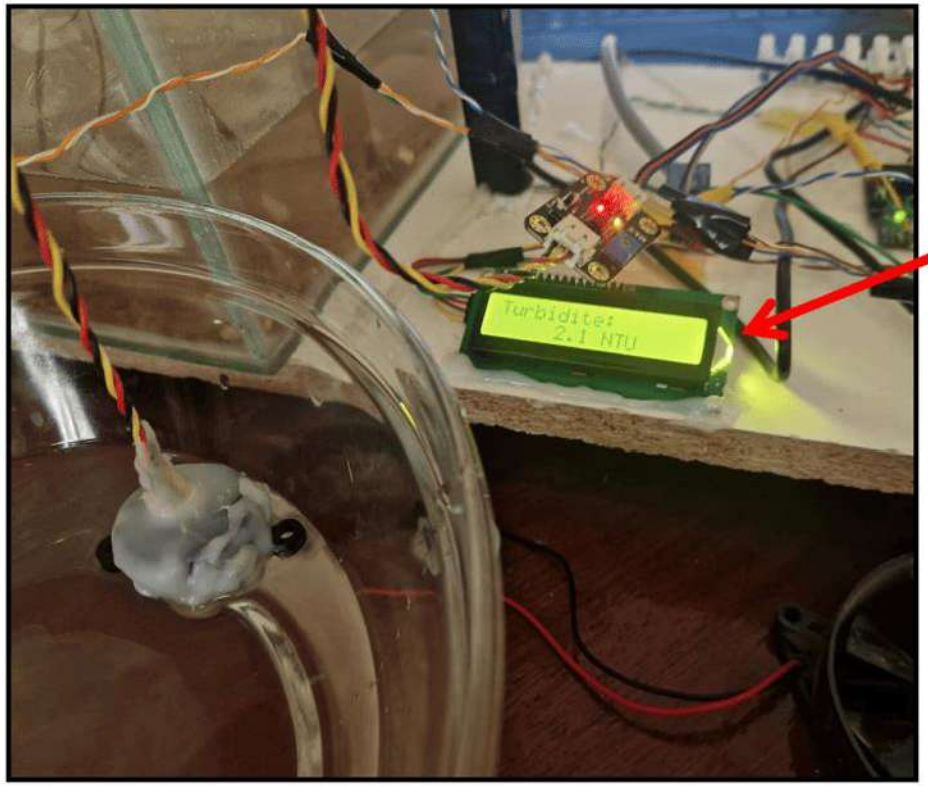


Expérience pour tester le filtre





Mesurer la turbidité

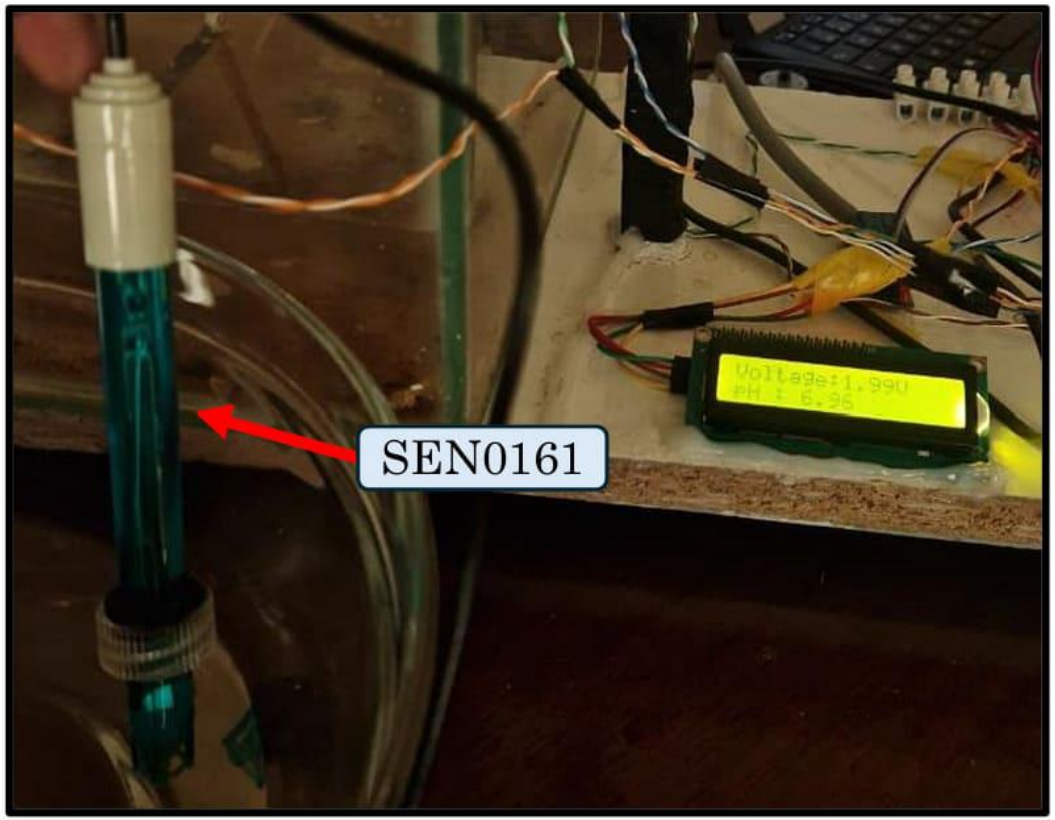


«requirement»
Régulation de la turbidité

Id = "1.1.2.2"
Text = "Le système doit maintenir la turbidité inférieur à 5 NTU"

Le résultat indique la valeur 2,1 NTU qui respect l'exigence

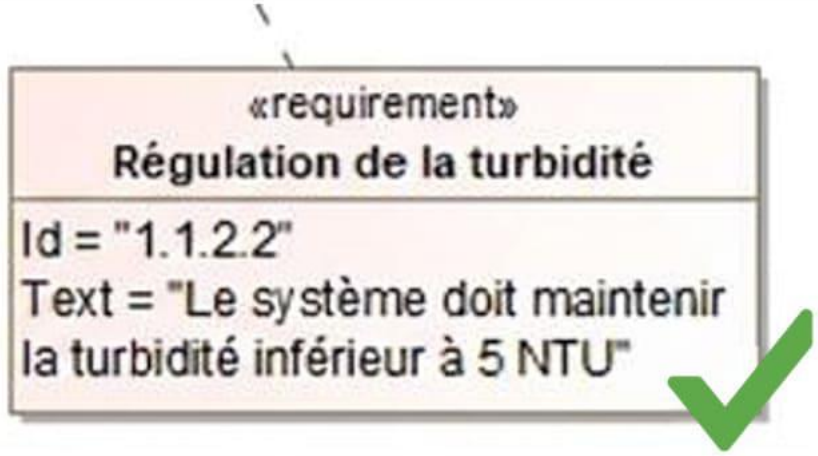
Mesurer le PH

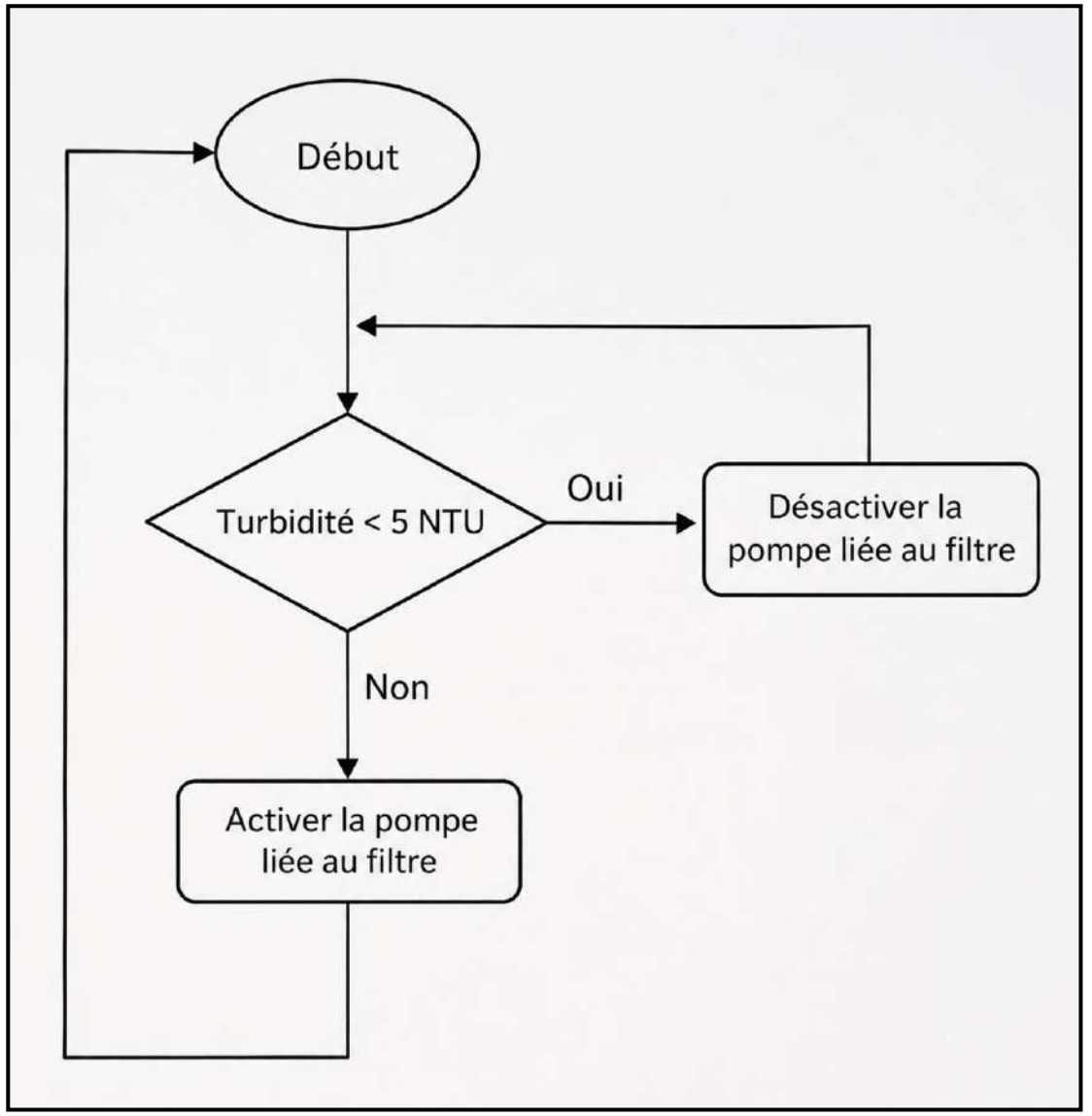


Le résultat indique la valeur 6,96 qui est compatible aux poissons

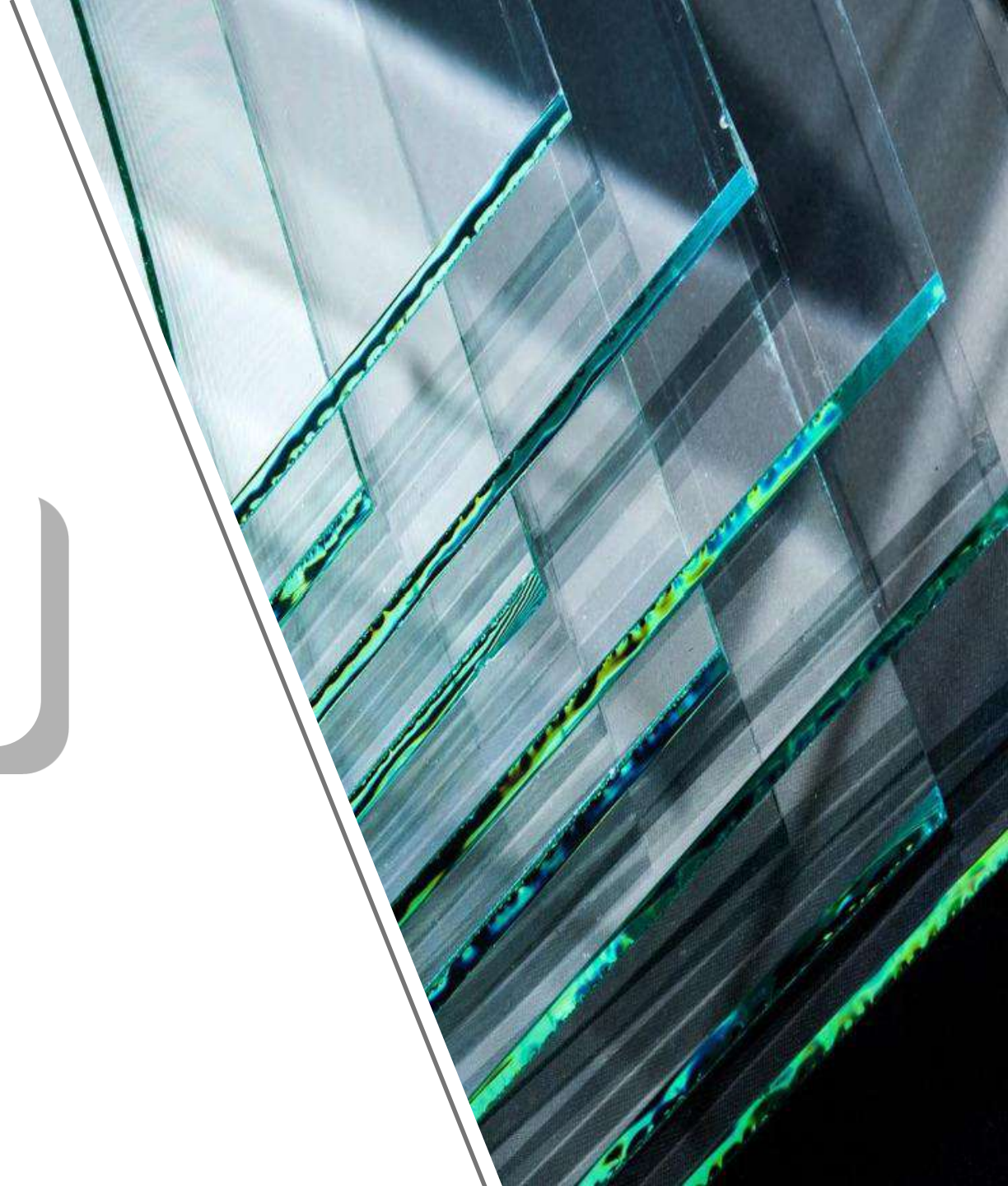
L’automatisation de la filtration :

La valeur de la turbidité ne doit pas dépasser **5 NTU** pour rester dans les conditions saines .





Objectif 3 :
Dimensionner et choix de matériau

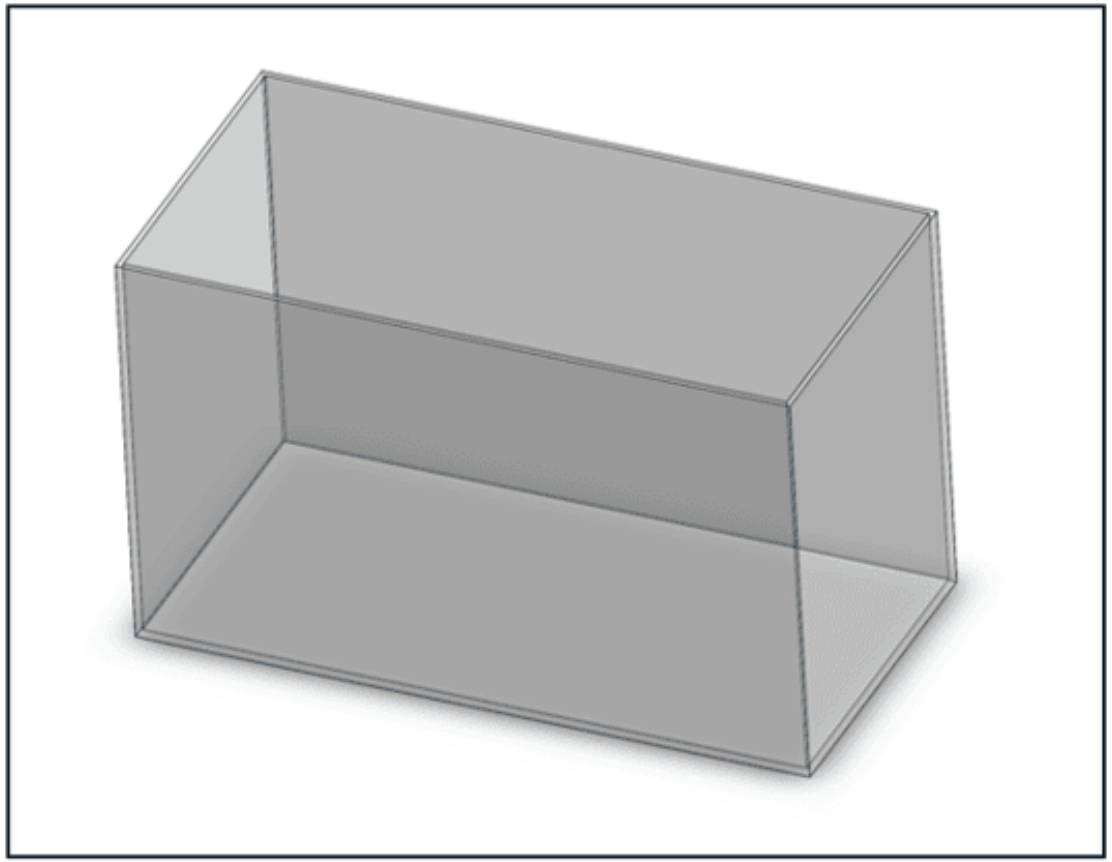


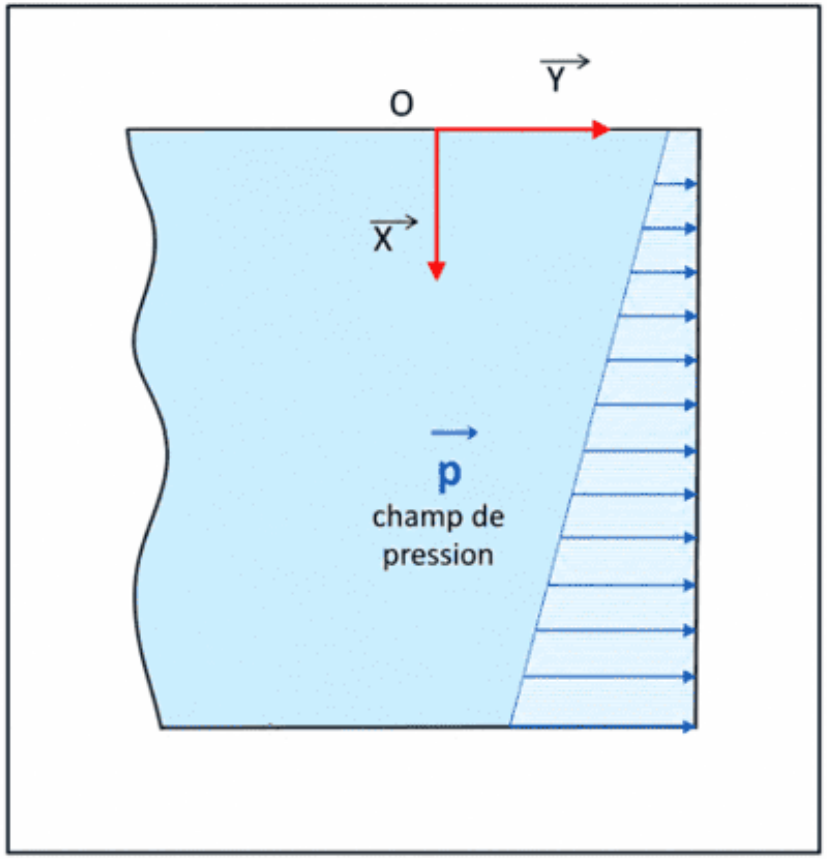
Dimensionner et choisir le matériau :

Pour accueillir un grand nombre de poissons ou des poissons de grande taille, il est souvent nécessaire d'utiliser un aquarium de grande taille. Cependant, le poids de l'eau génère des contraintes sur les parois en verre. Un mauvais choix du matériau ou de l'épaisseur du verre peut compromettre la résistance de la cuve.

Dimensionner et choisir le matériau :

Mise en évidence du **problème** étudié



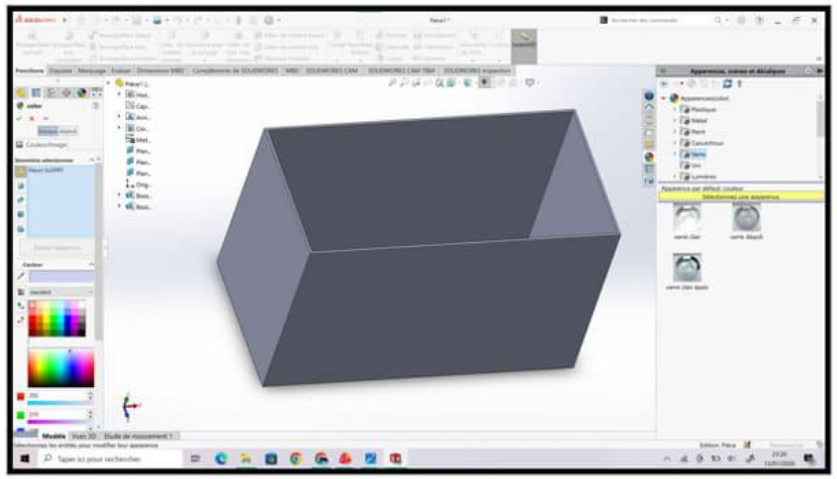


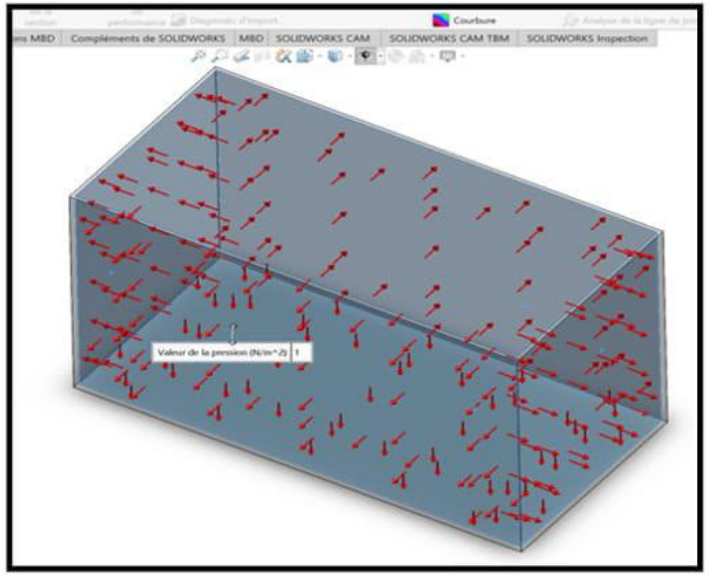
Expression du champ de la pression: $P(x) = \rho \cdot g \cdot x$

Dimensionner et choisir le matériau :

Modélisation du problème

SOLIDWORKS



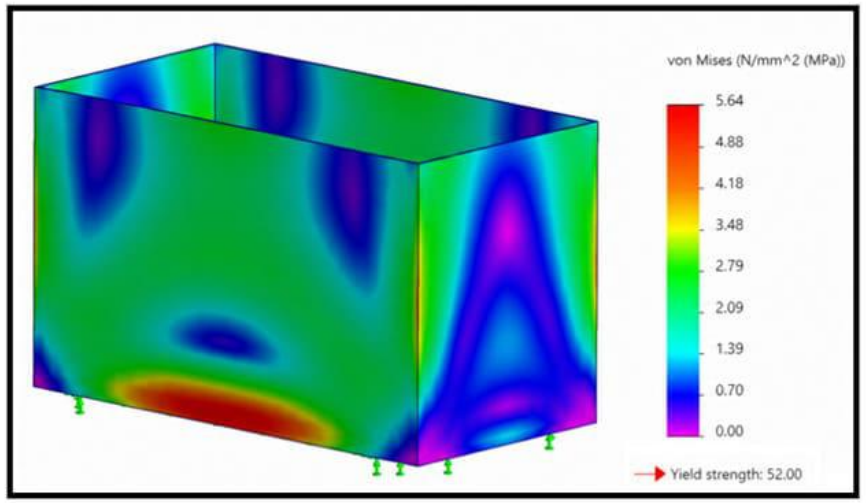


On a la pression de l'eau p(x) en (pa):

$$P(x) = \rho \cdot g \cdot x$$

AN : P(x) = 1000 * 9.81 * x = 9810 x

Après l'exécution on obtient :



Contrainte maximale = 5,64 Mpa
Limite d'élasticité: Re = 52 Mpa
Facteur de sécurité :s = 52/5.64 = 9, 21

Pour l'épaisseur de 4 mm la contrainte maximale obtenue est de 5,64 MPa, largement inférieure à la limite d'élasticité du matériau (52 MPa). La structure est donc mécaniquement résistante aux efforts appliqués.

Objectif 4 :

Réaliser une interface de communication avec l'utilisateur

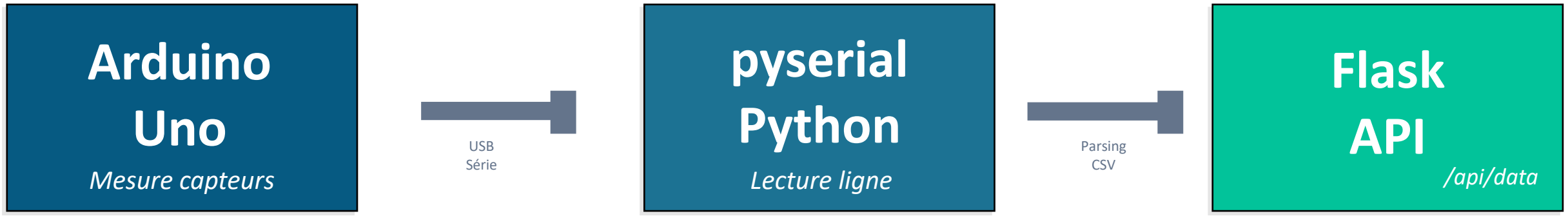
Architecture du système d'acquisition :

- Lecture port série avec pyserial
- Serveur web Flask sur le PC
- Dashboard HTML/CSS/JS en temps réel
- Alertes & export CSV intégrés
- Accès PC + téléphone

Technique utilisé

Python	Langage principal
pyserial	Lecture port série USB
Flask	Serveur web local
Chart.js	Graphiques temps réel
HTML / CSS	Interface utilisateur

Communication Arduino → Python :



Traitement local des signaux analogiques (Température/Turbidité) et filtrage numérique pour garantir la stabilité des mesures.

Il récupère le flux brut arrivant par le port USB et le décode pour rendre les informations lisibles par l'ordinateur.

Il centralise les données décodées et les rend disponibles instantanément pour le tableau de bord via une connexion réseau.

tableau de bord — Fonctionnalités :

Température Aquarium

25.0 °C

Température Réservoir

25.2 °C

Turbidité de l'eau

120 NTU

Graphiques Chart.js

Courbes temps réel — mise à jour toutes les 2 secondes

Alertes automatiques

Seuils min/max configurables directement sur le site

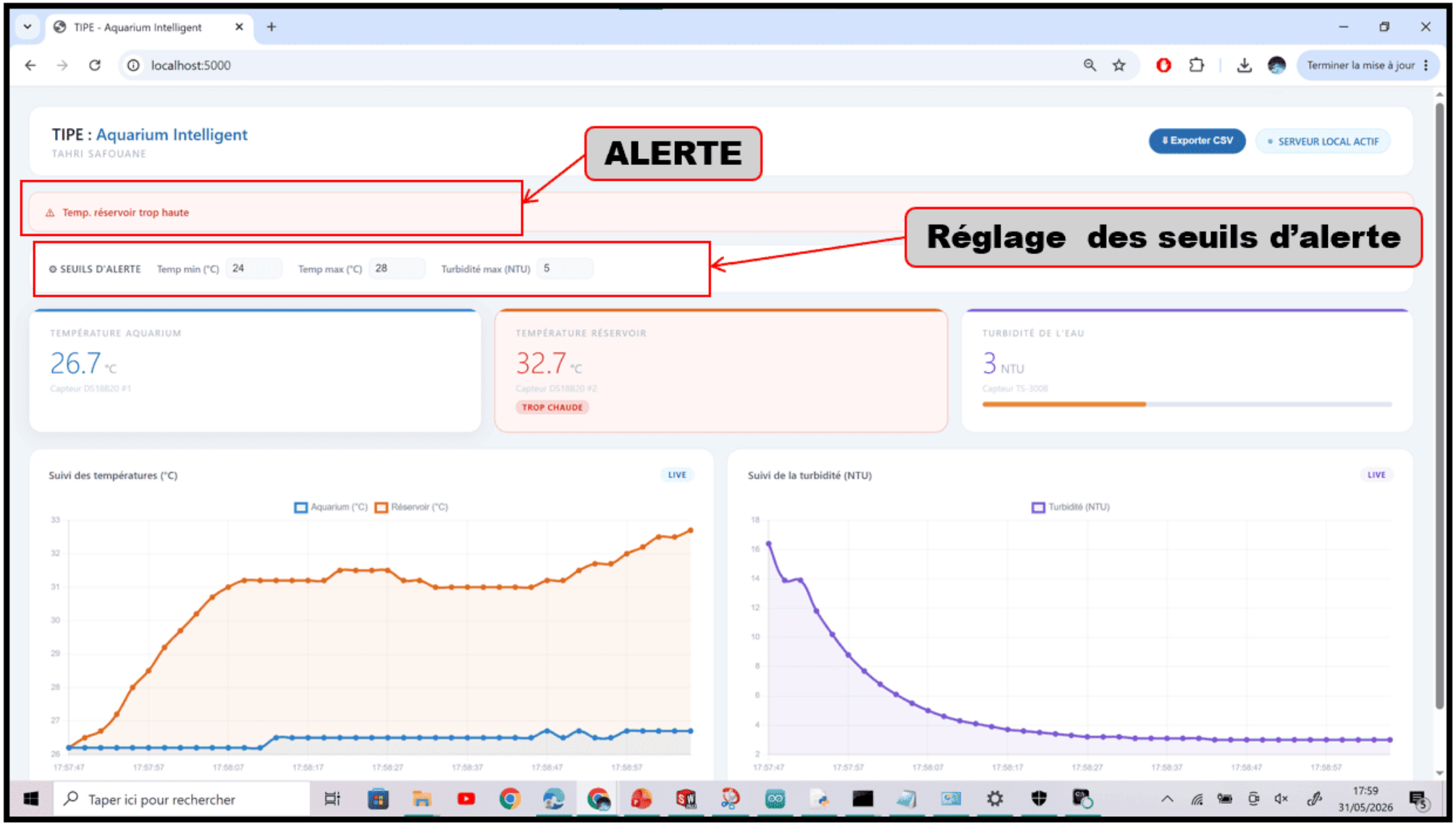
Export CSV

Téléchargement de tout l'historique de la session

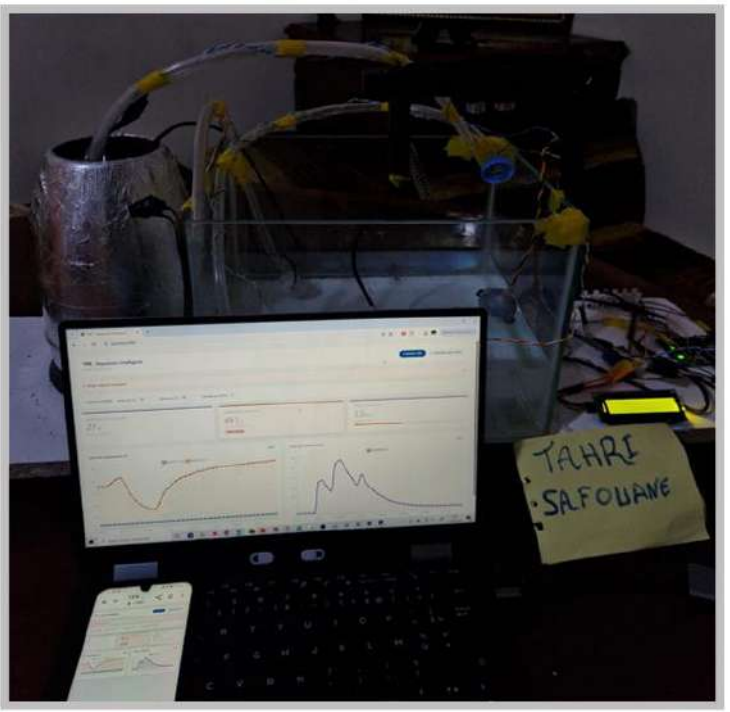
Accès multi-appareils

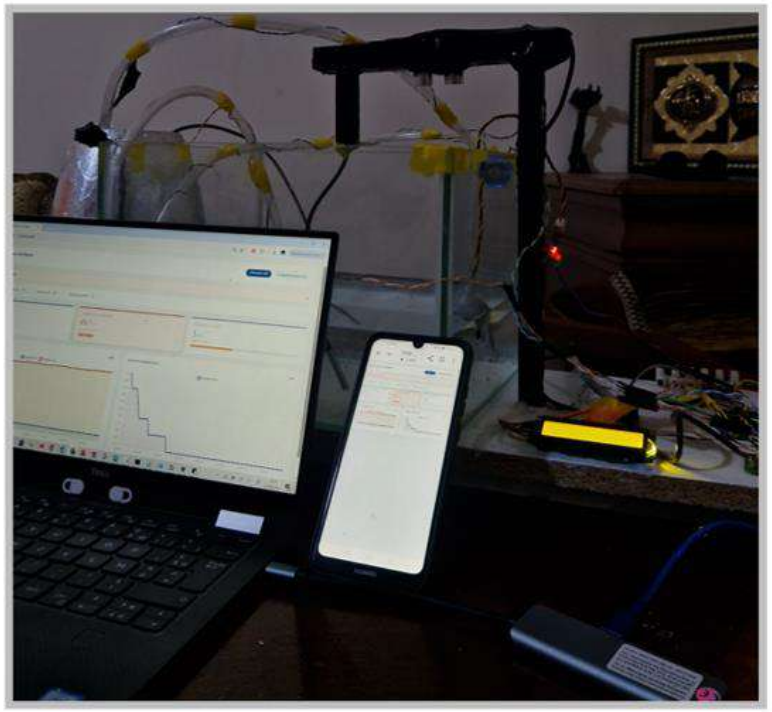
PC et téléphone sur le même réseau WiFi (192.168.1.66)

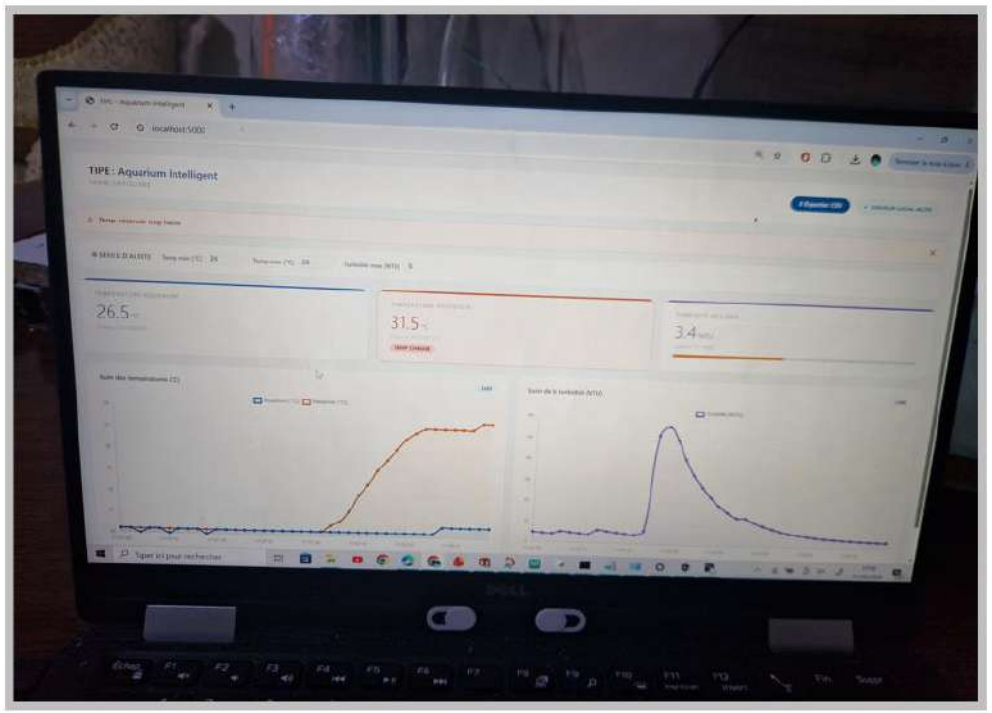
tableau de bord — Fonctionnalités :



RESULTAT :

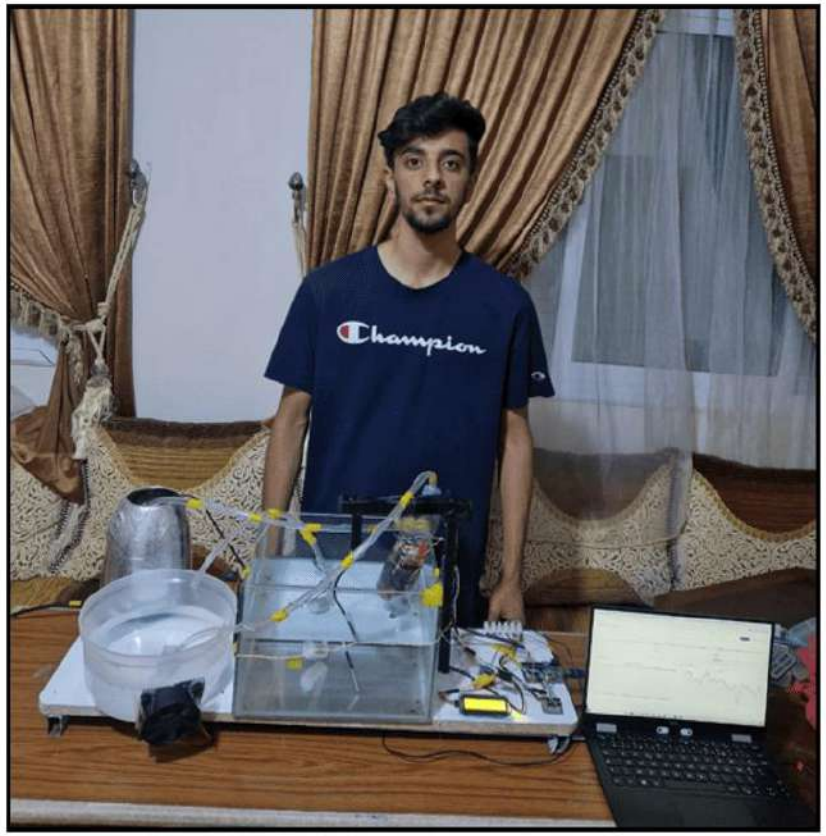






Objectif 5 :

Réaliser un prototype de l'aquarium


Champion

Merci pour votre attention

Annexes

❑ Code globale Arduino :

```ino
/*-------------------------------------------------------------------------*/
// Projet TIPE : Aquarium Intelligent - Correcteur PI
// Etudiant : TAHRI SAFOUANE
/*-------------------------------------------------------------------------*/

//-------------------------(Consignes)-------------------------------
const float Tc  = 28.0;  // Consigne température aquarium (°C)
const float Tch_MAX = 45.0;  // Sécurité chauffage
const float Tch_MIN = 43.0;  // Seuil activation chauffage

//-------------------------(Pins Pompes)-----------------------------
#define pc  9    // (réservoir chaud → aquarium)
#define pf  6    // (réservoir froid → aquarium)
#define ps  3    // (aquarium → réservoirs)

//-------------------------(Pins Relais)-----------------------------
#define Relais_Chauffage 12
#define Relais_Peltier   11

//-------------------------(LCD I2C)---------------------------------
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
int af = 0;
unsigned long tf = 0;

//-------------------------(DS18B20)---------------------------------
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
DeviceAddress cap_aq = {0x28, 0xB1, 0xE3, 0x57, 0x04, 0xE1, 0x3D, 0x88};
DeviceAddress cap_ch = {0x28, 0x94, 0x3A, 0x46, 0xD4, 0xCD, 0x1C, 0x27};

float Ta = 0, Tch = 0;
unsigned long tc_timer = 0;

//-------------------------(Turbidité)-------------------------------
#define TURB_PIN A1
float NTU = 3.0;
unsigned long t_turb = 0;

/*-------------------------------------------------------------------------*/
//                    Correcteur PI
/*-------------------------------------------------------------------------*/
const float Kp = 40.0;    // Gain proportionnel
const float Ki = 0.8;     // Gain intégral
const float dt = 0.5;     // Période d'échantillonnage (s) = 500ms

float integrale = 0;
float commande  = 0;

void correcteur_PI() {
  float erreur = Tc - Ta;

  if (!((commande >= 255 && erreur > 0) || (commande <= 0 && erreur < 0))) {
    integrale += erreur * dt;
  }

  commande = Kp * erreur + Ki * integrale;
  commande = constrain(commande, -255, 255);

  if (commande > 0) {
    // Aquarium trop froid → eau chaude
    analogWrite(pc, (int)commande);   // Pompe chaude proportionnelle
    analogWrite(ps, (int)commande);   // Soutirage proportionnel
    analogWrite(pf, 0);
  }
  else if (commande < 0) {
    // Aquarium trop chaud → eau froide
    int cmd = (int)(-commande);
    analogWrite(pf, cmd);             // Pompe froide proportionnelle
    analogWrite(ps, cmd);             // Soutirage proportionnel
    analogWrite(pc, 0);
  }
  else {
    // Erreur nulle → tout OFF
    analogWrite(pc, 0);
    analogWrite(pf, 0);
    analogWrite(ps, 0);
  }
}

/*-------------------------------------------------------------------------*/
//                    Relais Chauffage (couplé à pc)
/*-------------------------------------------------------------------------*/
void chauffage() {
  if (Tch > Tch_MAX) {
    digitalWrite(Relais_Chauffage, HIGH);
    return;
  }

  if (commande > 0) {
    digitalWrite(Relais_Chauffage, Tch < Tch_MIN ? LOW : HIGH);
  } else {
    digitalWrite(Relais_Chauffage, HIGH); // Inutile de chauffer
  }
}

/*-------------------------------------------------------------------------*/
//                    Relais Peltier (couplé à Ta)
/*-------------------------------------------------------------------------*/
void peltier() {
```

aquarium_PI.ino

```
106
107   digitalWrite(Relais_Peltier, commande < -10 ? LOW : HIGH);
108 }
109
110 /*-------------------------------------------------------------------------*/
111 //                    Turbidité (TS-300B)
112 /*-------------------------------------------------------------------------*/
113 float readVoltageFiltered() {
114   float v[21];
115   for (int i = 0; i < 21; i++) { v[i] = analogRead(TURB_PIN) * 5.0 / 1023.0; delay(5); }
116
117   for (int i = 0; i < 20; i++)
118     for (int j = i+1; j < 21; j++)
119       if (v[j] < v[i]) { float t = v[i]; v[i] = v[j]; v[j] = t; }
120   return v[10];
121 }
122
123 void turbidite_cap() {
124   if (millis() - t_turb >= 1000) {
125     t_turb = millis();
126     float voltage = readVoltageFiltered();
127     float turb;
128     if      (voltage >= 4.20) turb = 3.0;
129     else if (voltage <= 2.50) turb = 1000.0;
130     else    turb = constrain((4.20 - voltage) / 1.70 * 1000.0 - 300.0, 3.0, 1000.0);
131     NTU = NTU * 0.90 + turb * 0.10; // Filtre passe-bas
132   }
133 }
134
135 /*-------------------------------------------------------------------------*/
136 //                    Affichage LCD + Série Flask
137 /*-------------------------------------------------------------------------*/
138 void affichage() {
139   if (millis() - tf >= 3000) {
140     tf = millis();
```

Output

aquarium_PI.ino

```
141     lcd.clear();
142     if (af == 0) {
143       lcd.setCursor(0, 0); lcd.print("Aq:"); lcd.print(Ta, 1); lcd.print((char)223); lcd.print("C");
144       lcd.setCursor(0, 1); lcd.print("Ch:"); lcd.print(Tch,1); lcd.print((char)223); lcd.print("C");
145     }
146     else if (af == 1) {
147       lcd.setCursor(0, 0); lcd.print("Turb:"); lcd.print(NTU, 1); lcd.print(" NTU");
148       lcd.setCursor(0, 1); lcd.print("Cmd PI:"); lcd.print((int)commande);
149     }
150     af = (af + 1) % 2;
151   }
152
153   // Série Flask : Ta, Tch, NTU, commande PI, Peltier, Chauffage
154   static unsigned long t_s = 0;
155   if (millis() - t_s >= 2000) {
156     t_s = millis();
157     Serial.print(Ta,1);    Serial.print(",");
158     Serial.print(Tch,1);   Serial.print(",");
159     Serial.print(NTU,1);   Serial.print(",");
160     Serial.print((int)commande); Serial.print(",");
161     Serial.print(digitalRead(Relais_Peltier)   == LOW ? 1 : 0); Serial.print(",");
162     Serial.println(digitalRead(Relais_Chauffage) == LOW ? 1 : 0);
163   }
164 }
165
166 /*-------------------------------------------------------------------------*/
167 //                    Setup
168 /*-------------------------------------------------------------------------*/
169 void setup() {
170   Serial.begin(9600);
171   sensors.begin();
172   sensors.setResolution(cap_aq, 10);
173   sensors.setResolution(cap_ch, 10);
174
175   pinMode(Relais_Chauffage, OUTPUT); digitalWrite(Relais_Chauffage, HIGH);
```

Output

```
176   pinMode(Relais_Peltier,   OUTPUT); digitalWrite(Relais_Peltier,   HIGH);
177   pinMode(pc, OUTPUT); analogWrite(pc, 0);
178   pinMode(pf, OUTPUT); analogWrite(pf, 0);
179   pinMode(ps, OUTPUT); analogWrite(ps, 0);
180
181   lcd.init(); lcd.backlight();
182   delay(1500); lcd.clear();
183 }
184
185 /*-------------------------------------------------------------------------*/
186 //                    Boucle Principale
187 /*-------------------------------------------------------------------------*/
188 void loop() {
189   if (millis() - tc_timer >= 500) {
190     tc_timer = millis();
191     sensors.requestTemperatures();
192     Ta  = sensors.getTempC(cap_aq);
193     Tch = sensors.getTempC(cap_ch);
194
195     correcteur_PI();
196     chauffage();
197     peltier();
198   }
199
200   turbidite_cap();
201   affichage();
202 }
203
```

Output

code Python et HTML : Passerelle de données et Interface de pilotage

1

2

3

4

passerelle_local.py - C:\Users\MomoPC\Desktop\passerelle_local.py (3.13.3)

File Edit Format Run Options Window Help

```
    cardT.classList.remove('alert-card'); tagT.classList.remove('show');
  }

  const banner = document.getElementById('alert-banner');
  if (msgs.length > 0) {
    document.getElementById('alert-text').textContent = msgs.join(' | ');
    banner.classList.add('show');
  } else {
    banner.classList.remove('show');
  }
}

function closeAlert() { document.getElementById('alert-banner').classList.remove('show'); }

function exportCSV() {
  if (allHistory.length === 0) { alert('Pas encore de données.'); return; }
  let csv = 'Heure,Temp Aquarium (°C),Temp Réservoir (°C),Turbidité (NTU)\\n';
  allHistory.forEach(h => { csv += h.time + ',' + h.temp1 + ',' + h.temp2 + ',' + h.turbidity + '\\n'; });
  const blob = new Blob([csv], { type: 'text/csv;charset=utf-8;' });
  const url  = URL.createObjectURL(blob);
  const a    = document.createElement('a');
  a.href = url;
  a.download = 'aquarium_export.csv';
  a.click();
  URL.revokeObjectURL(url);
}

async function update() {
  try {
    const res = await fetch('/api/data');
    const d   = await res.json();
    document.getElementById('t1').innerHTML   = d.temp1 + '<span class="unit">°C</span>';
    document.getElementById('t2').innerHTML   = d.temp2 + '<span class="unit">°C</span>';
    document.getElementById('turb').innerHTML = d.turbidity + '<span class="unit">NTU</span>';
    allHistory = d.history;
    checkAlerts(d);
    updateGauge(d.turbidity);
    const labels = d.history.map(h => h.time);
    chartTemp.data.labels = labels;
    chartTemp.data.datasets[0].data = d.history.map(h => h.temp1);
    chartTemp.data.datasets[1].data = d.history.map(h => h.temp2);
    chartTemp.update();
    chartTurb.data.labels = labels;
    chartTurb.data.datasets[0].data = d.history.map(h => h.turbidity);
    chartTurb.update();
  } catch(e) {}
}

setInterval(update, 2000);
update();
</script>
</body>
</html>"""
```

5

```python
def read_serial():
    try:
        ser = serial.Serial(SERIAL_PORT, 9600, timeout=2)
        while True:
            line = ser.readline().decode('utf-8', errors='ignore').strip()
            if ',' in line:
                parts = line.split(',')
                if len(parts) == 3:
                    try:
                        t1, t2, turb = float(parts[0]), float(parts[1]), float(parts[2])
                        latest_data["temp1"] = round(t1, 1)
                        latest_data["temp2"] = round(t2, 1)
                        latest_data["turbidity"] = round(turb, 1)
                        latest_data["history"].append({
                            "time": time.strftime("%H:%M:%S"),
                            "temp1": round(t1, 1),
                            "temp2": round(t2, 1),
                            "turbidity": round(turb, 1)
                        })
                        if len(latest_data["history"]) > 40: latest_data["history"].pop(0)
                    except: pass
    except: pass

@app.route('/api/data')
def api_data(): return jsonify(latest_data)

@app.route('/')
def index(): return render_template_string(HTML_PAGE)

if __name__ == '__main__':
    threading.Thread(target=read_serial, daemon=True).start()
    app.run(host='0.0.0.0', port=5000)
```

6