Régulation de température d'une cuve (Groupe B)

Informations :

Noms et prénoms :

SEJDI Linda (linda.sejdi@etu.sorbonne-universite.fr)
SOSA VARGAS Luis Angel (la.sosa.vargas@gmail.com)
THAVARAJAH Roshani (roshani.thavarajah@etu.sorbonne-universite.fr)
VASANTHAN Luxcheni (luxcheni.vasanthan@etu.sorbonne-universite.fr)

Période : Octobre 2024 - Janvier 2025

Cursus : Master 2 Chimie parcours Ingénierie Chimique

Tuteur : PULPYTEL Jerome (jerome.pulpytel@sorbonne-universite.fr)

Contexte :

Ce projet, réalisé dans le cadre de l'UE 803 "Optimisation et Contrôle des Procédés", porte sur la régulation et l'automatisation de la température d'une cuve contenant un liquide. Ce projet vise à développer des compétences pratiques en régulation, automatisation et intégration électronique.

Objectifs :

L'objectif de ce projet est de concevoir et de réaliser un système automatisé de régulation de température d'une cuve. Il s'appuie sur une carte Arduino, un module de Peltier, une sonde de température et un circuit électronique intégrant un écran LCD, une LED et un buzzer. Ces éléments permettent de mesurer, d'afficher en temps réel et de réguler automatiquement la température pour atteindre une consigne définie par l'utilisateur. Le système devra être capable d'assurer les fonctions suivantes :

1. **Imposer une température cible :**
   La température cible est fixée dans le code Arduino au démarrage. Cette consigne initiale sert de référence pour la régulation.

2. **Afficher en temps réel :**
   La température mesurée par une sonde immergée dans la cuve est affichée en temps réel

sur un écran LCD. Cela permet une visualisation claire de l'évolution de la température.

3. **Réguler via le module de Peltier :**
   Le module de Peltier est chargé de chauffer ou de refroidir la cuve pour maintenir la température souhaitée.
   Une LED RGB indique visuellement l'état de la régulation. Suivant la couleur on a des informations sur l'état de la température.
   - Rouge si la température est inférieure à la consigne
   - Bleue si la température est supérieure à la consigne
   - Vert si la température correspond à la consigne

4. **Ajuster dynamiquement la température cible :**
   Deux boutons poussoirs permettent à l'utilisateur de modifier la température cible en l'augmentant ou en la diminuant.
   L'écran LCD met immédiatement à jour la nouvelle consigne et le système reprend sa régulation jusqu'à ce que la nouvelle température cible soit atteinte.

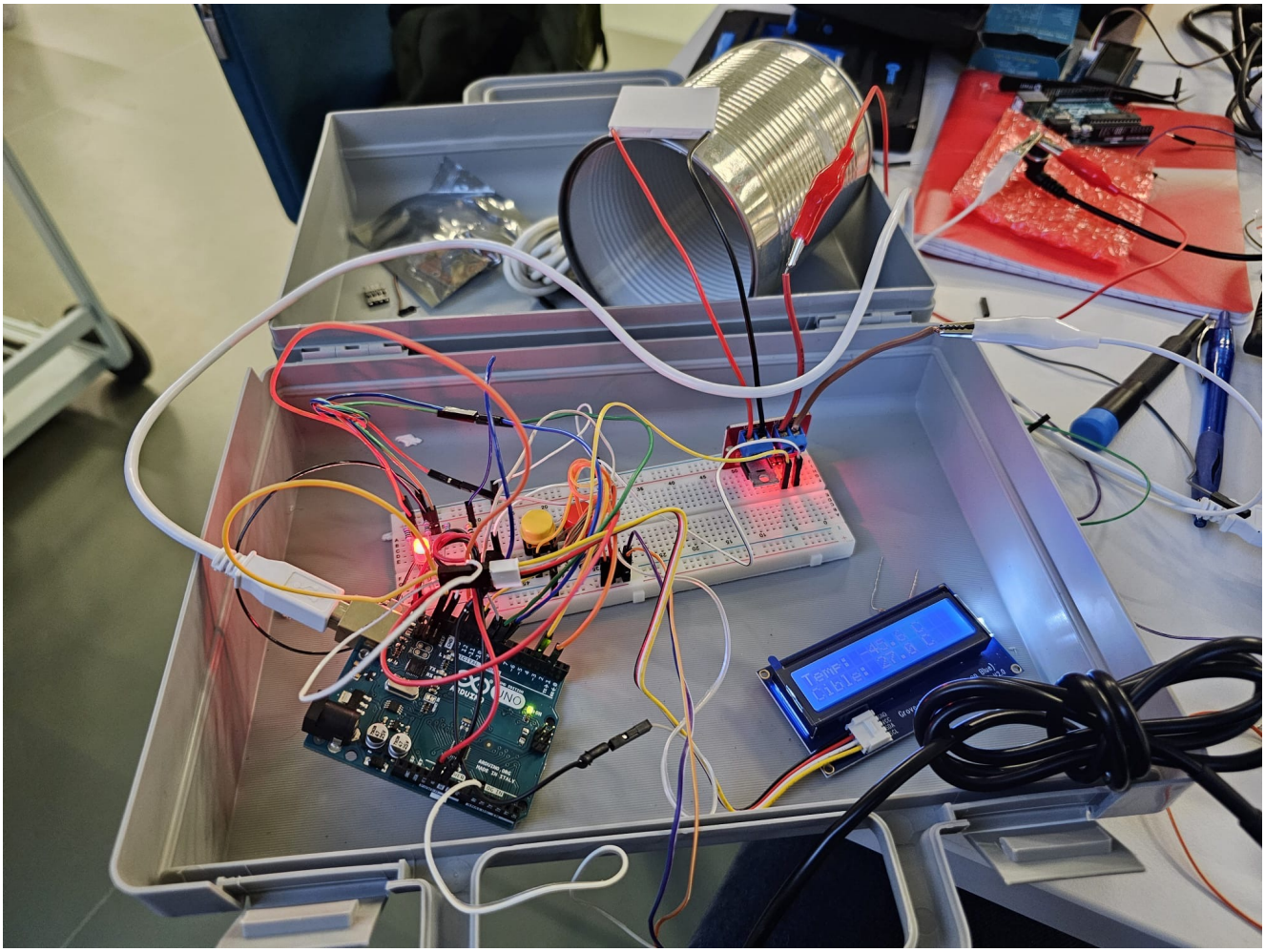


Figure 1 : Montage du système de régulation thermique avec affichage LCD et le module Peltier

Matériel :

Catégorie	Matériaux	Quantité	Rôles = Fonctions
Carte de contrôle	Carte Arduino UNO	1	Microcontrôleur pour gérer tout le système.
Chauffage ou refroidissement	Module Peltier	1	Chauffer ou refroidir selon le courant qui traverse.
	Module MOS (driver de puissance)	1	Contrôler la puissance délivrée au Module Peltier.
Capteur de température	Sonde de température (sonde DS1)	1	Mesurer la température du liquide contenu dans la cuve.
	Résistances (100 Ω, 220 Ω, 4,7 kΩ et 10 kΩ)	1	Assurer une communication stable avec les capteurs.
Connexions	Câble Dupont	Plusieurs	Assurer les connexions entre les différents composants.
	Câble USB pour Arduino	1	Permet de téléverser le programme et d'alimenter la carte Arduino.
	Breadboard (plaque de prototypage)	1	Permettre les montages électroniques sans soudure.
	Pâte thermique	1	Améliorer le transfert thermique entre les surfaces.
Alimentation	Alimentation externe de 12V (adapteur secteur ou batterie)	1	Fournir l'énergie nécessaire au Module Peltier.
Affichage (optionnel)	Ecran LCD (16X2)	1	Afficher la température en temps réel.
Indicateurs	LED RGB	1	Indiquer les état de température (froid, normal, chaud).
Cuve et fixation	Cuve en aluminium	1	Contenir le liquide dont la température est régulée.
	Support pour la cuve	1	Stabiliser la cuve.
	Pompe de circulation	1	Faire circuler le liquide pour uniformiser la température.
	Pince en C	1	Fixer solidement les composants mécaniques.
Outils et accessoires	Fer à souder et étain	1	Souder les connexions de manière permanente si besoin.
	Multimètre	1	Vérifier les tensions.
	Tournevis, pinces et visseries	Plusieurs	Assembler et fixer les composants mécaniques.
Composants électroniques	Transistor NMOS (IRF540N ou similaire)	1	Permettre le contrôle en puissance du Module Peltier.
Commandes supplémentaires	Boutons poussoirs	2	Permettre l'activation ou la désactivation manuelle du système.
	Buzzer	1	Signaler un état ou un avertissement.

Connexions :

Module Peltier :

- Pôle + : Alimentation externe
- Pôle - : Source du transistor

Sonde DS1 :

- VCC : Alimentation 5V
- GND : GND
- Signal : Broche D2 de l'Arduino
- Résistance de 4,7 kΩ en série avec le signal

Transistor NMOS (IRF540N ou similaire) :

- Source : GND
- Drain : Circuit du module Peltier
- Gate : Broche D12 de l'Arduino
- Drain relié à une borne du module Peltier, l'autre borne reliée au +12V de l'alimentation externe
- Résistance de 220 Ω entre la Gate et le sortie D12

LED RGB :

- Cathode : Commune à GND
- Broches rouge, verte et bleue : connectées respectivement à D10, D9 et D8 de l'Arduino, via des résistances de 220 Ω

Ecran LCD 16x2 :

- SCL : Broche A5 de l'Arduino
- SDA : Broche A4 de l'Arduino
- VCC : +5V de l'Arduino
- GND : GND de l'Arduino

Buzzer :

- Buzzer : Broche D13 de l'Arduino
- Résistance de 100Ω en série avec le buzzer

Bouton poussoir :

- Une broche : +5V
- Résistance de pull-down de 10 kΩ entre la broche et GND

Construction :

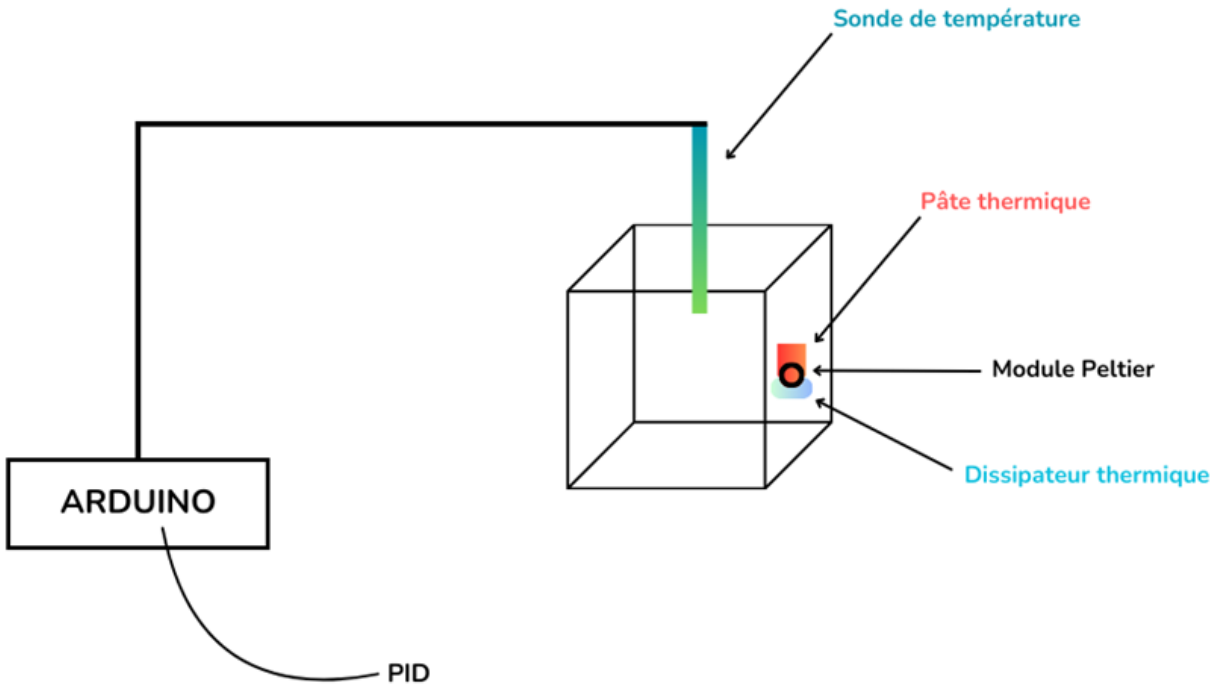


Figure 2 : Schéma du système de régulation de température avec Arduino et module Peltier

Étape 1 : Conception et préparation

La première étape consiste à bien comprendre le sujet et à établir une liste détaillée des composants nécessaires. Cette phase inclut également l'organisation des tâches au sein de l'équipe, chaque membre se voyant attribuer des responsabilités spécifiques.

Étape 2 : Montage et assemblage

La deuxième étape implique la réception du matériel et l'assemblage des composants. L'équipe installe le kit de refroidissement Peltier sur la cuve et relie tous les éléments électroniques sur une platine d'essai.

Étape 3 : Programmation et test

La troisième étape est dédiée au développement du code pour l'Arduino UNO, qui contrôle le module Peltier et régule la température de la cuve. Le programme inclut l'implémentation d'un régulateur PID pour assurer un contrôle précis de la température.

Programme Arduino

```
#include <Wire.h>               // Pour la communication I2C
#include <LiquidCrystal_I2C.h>   // Pour l'écran LCD I2C
#include <OneWire.h>            // Pour la communication avec le capteur DS18B20
#include <DallasTemperature.h>   // Pour lire la température du capteur DS18B20
// Définition des broches
#define ONE_WIRE_BUS 2          // Broche pour le capteur DS18B20
#define BTN_UP_PIN 3            // Bouton poussoir pour augmenter la température de consigne
#define BTN_DOWN_PIN 4          // Bouton poussoir pour diminuer la température de consigne
#define LED_R_PIN 10            // Broche pour la LED RGB rouge
#define LED_G_PIN 9             // Broche pour la LED RGB verte
#define LED_B_PIN 8             // Broche pour la LED RGB bleue
#define MOTOR_PIN 12            // Broche pour contrôler le module Peltier
#define BUZZER_PIN 13           // Broche pour le buzzer
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Initialisation de l'écran LCD I2C (adresse: 0x27)
OneWire oneWire(ONE_WIRE_BUS);       // Initialisation de la communication OneWire
DallasTemperature sensors(&oneWire); // Initialisation du capteur DS18B20
int targetTemp = 27; // Température cible initiale
float currentTemp;
void setup() {
  // Initialisation de l'écran LCD I2C
  lcd.begin();
  lcd.backlight();  // Activer le rétroéclairage
  // Afficher un message d'accueil
  lcd.setCursor(0, 0);
  lcd.print("Regulation Temp");
  delay(2000);
  lcd.clear();
  // Configuration des broches
  pinMode(BTN_UP_PIN, INPUT_PULLUP);
  pinMode(BTN_DOWN_PIN, INPUT_PULLUP);
  pinMode(LED_R_PIN, OUTPUT);
  pinMode(LED_G_PIN, OUTPUT);
  pinMode(LED_B_PIN, OUTPUT);
  pinMode(MOTOR_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  // Initialisation du capteur de température
  sensors.begin();
}
void loop() {
  // Lire la température actuelle du DS18B20
  sensors.requestTemperatures(); // Demande de mise à jour de la température
  currentTemp = sensors.getTempCByIndex(0); // Obtenir la température en Celsius
  // Afficher la température actuelle sur l'écran LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(currentTemp);
  lcd.print(" C    "); // Espaces pour effacer les caractères restants
  // Gestion des boutons poussoirs pour ajuster la température cible
  if (digitalRead(BTN_UP_PIN) == LOW) {
    targetTemp++;
    delay(200);  // Anti-rebond
```

```
  }
  if (digitalRead(BTN_DOWN_PIN) == LOW) {
    targetTemp--;
    delay(200);  // Anti-rebond
  }
  // Afficher la température cible sur la deuxième ligne de l'écran LCD
  lcd.setCursor(0, 1);
  lcd.print("Cible: ");
  lcd.print(targetTemp);
  lcd.print(" C   ");
  // Gestion du module Peltier et des LEDs en fonction de la température
  if (currentTemp < targetTemp) {
    digitalWrite(MOTOR_PIN, HIGH); // Active le module Peltier
    digitalWrite(LED_R_PIN, HIGH); // Allumer la LED rouge
    digitalWrite(LED_G_PIN, LOW);
    digitalWrite(LED_B_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW); // Ne pas activer le buzzer
  } else if (currentTemp > targetTemp) {
    digitalWrite(MOTOR_PIN, LOW); // Désactive le module Peltier
    digitalWrite(LED_R_PIN, LOW);
    digitalWrite(LED_G_PIN, LOW);
    digitalWrite(LED_B_PIN, HIGH); // Allumer la LED bleue (température dépassée)
    digitalWrite(BUZZER_PIN, LOW); // Ne pas activer le buzzer
  } else {
    digitalWrite(MOTOR_PIN, LOW); // Désactive le module Peltier
    digitalWrite(LED_R_PIN, LOW);
    digitalWrite(LED_G_PIN, HIGH); // Allumer la LED verte (température atteinte)
    digitalWrite(LED_B_PIN, LOW);
    digitalWrite(BUZZER_PIN, HIGH); // Activer le buzzer
    delay(1000); // Sonner une seconde
    digitalWrite(BUZZER_PIN, LOW); // Éteindre le buzzer
  }
  delay(500); // Pause pour stabiliser les lectures et les réactions
}
```

Journal de bord :

07/10/2024 :

- Attribution du projet de régulation de la température d'une cuve.
- Visite du FabLab pour découvrir les ressources disponibles.
- Réflexion sur la méthode de régulation à utiliser et prise de contact avec le groupe pour organiser la première réunion.

11/10/2024 :

- Première réunion Zoom entre les membres de l'équipe.
- Comparaison et validation de la liste finale de matériel.

17/10/2024 :

- Rendez-vous avec la tutrice pour valider officiellement la liste de matériel.
- Récupération du matériel.
- Commande du module de Peltier, dont la livraison est prévue pour mi-novembre.
- **Prochaine étape** : Simulation du système sur Tinkercad pour tester le concept avant la mise en œuvre physique.

03/11/2024 :

- Deuxième réunion Zoom entre les membres de l'équipe.
- Comparaison des simulations réalisées sur Tinkercad.
- Demande d'avis au tuteur sur le montage et le code :

  Lorsque nous avons présenté notre simulation à notre tuteur, nous lui avons expliqué que nous rencontrions un problème avec les boutons poussoirs qui ne semblaient pas fonctionner. Nous lui avons demandé des remarques et suggestions pour nous aider à avancer. En réponse, il nous a précisé que les boutons fonctionnaient correctement. Lorsqu'on appuie sur le bouton, la tension passe bien de 4,17 V à 0 V. Cependant, il a souligné que le problème venait des fonctions "delay" et du temps de simulation non réel. Il est nécessaire de maintenir le bouton enfoncé pendant environ 3 à 4 secondes pour que l'incrément de température soit pris en compte.

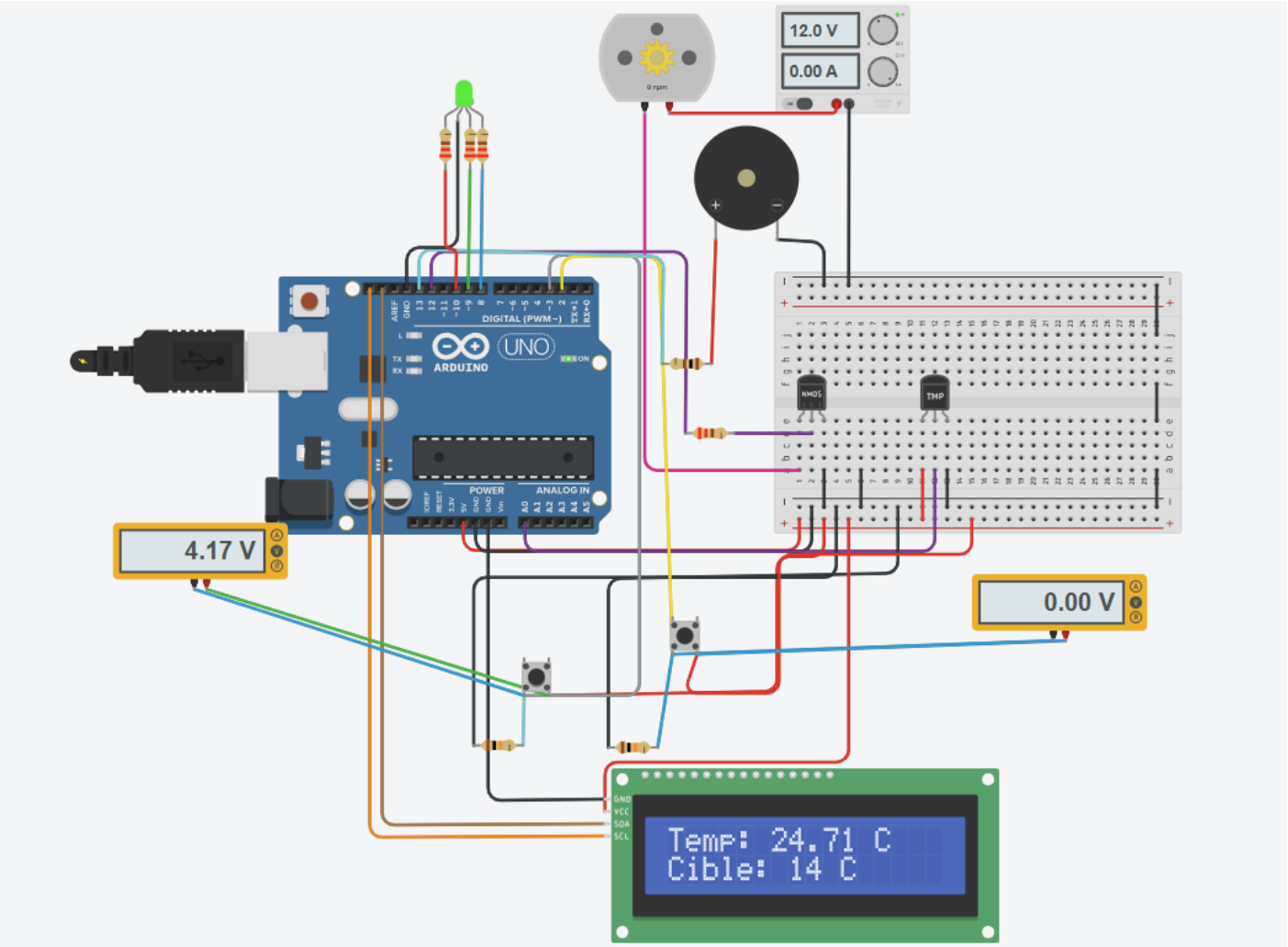


Figure 3 : Simulation du système sur Tinkercad

07/11/2024 :

- Récupération du matériel manquant que l'on avait commandé : Module Peltier, Transistor MOS, Breadboard, Pâte thermique, Ecran LCD

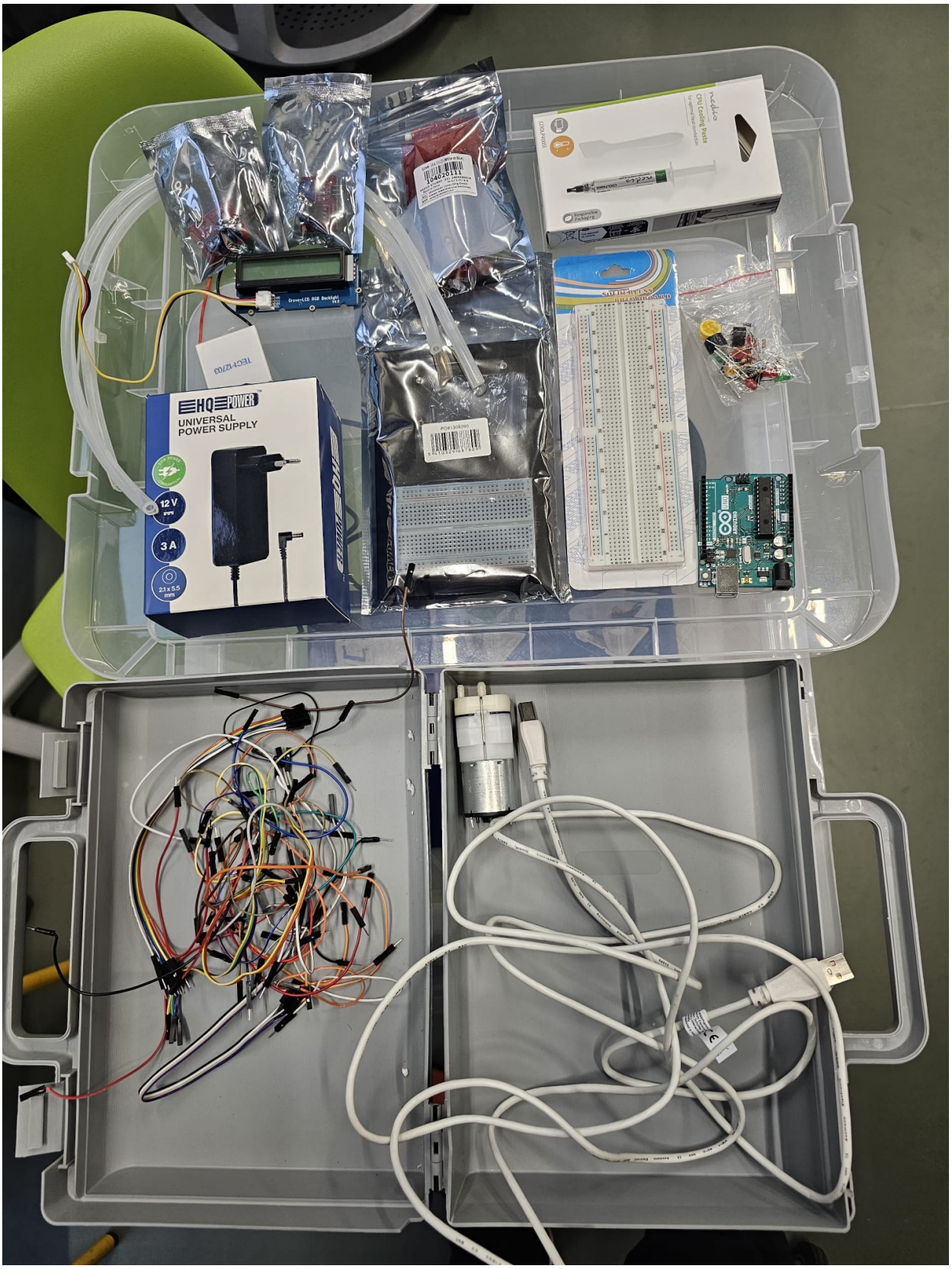
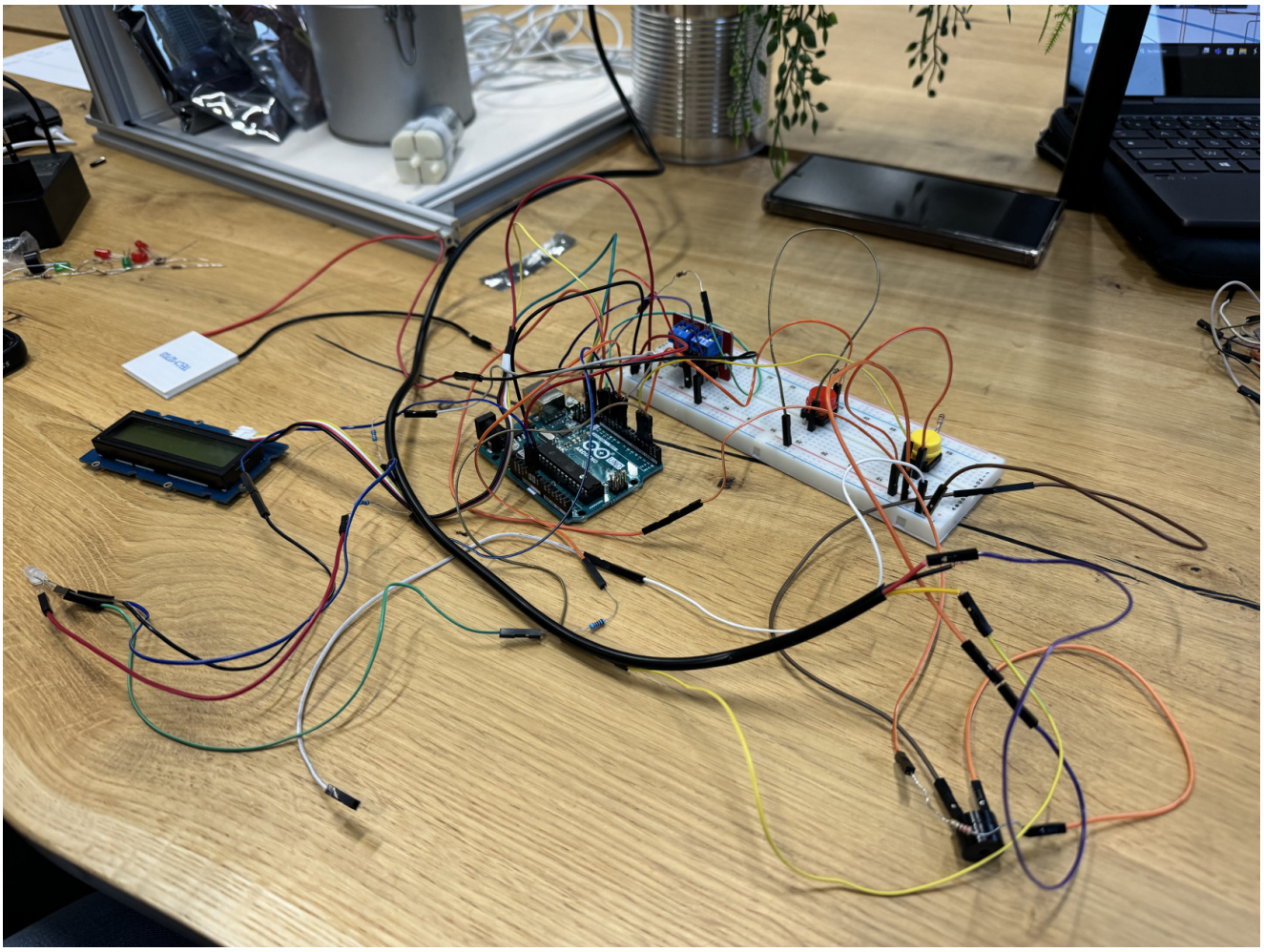


Figure 4 : Matériel complet prêt pour l'assemblage du projet

21/11/2024 :

- Première réunion au FabLab.

- Nous avons débuté l'assemblage du montage en suivant précisément la simulation réalisée sur Tinkercard.
- Problème de port identifié.



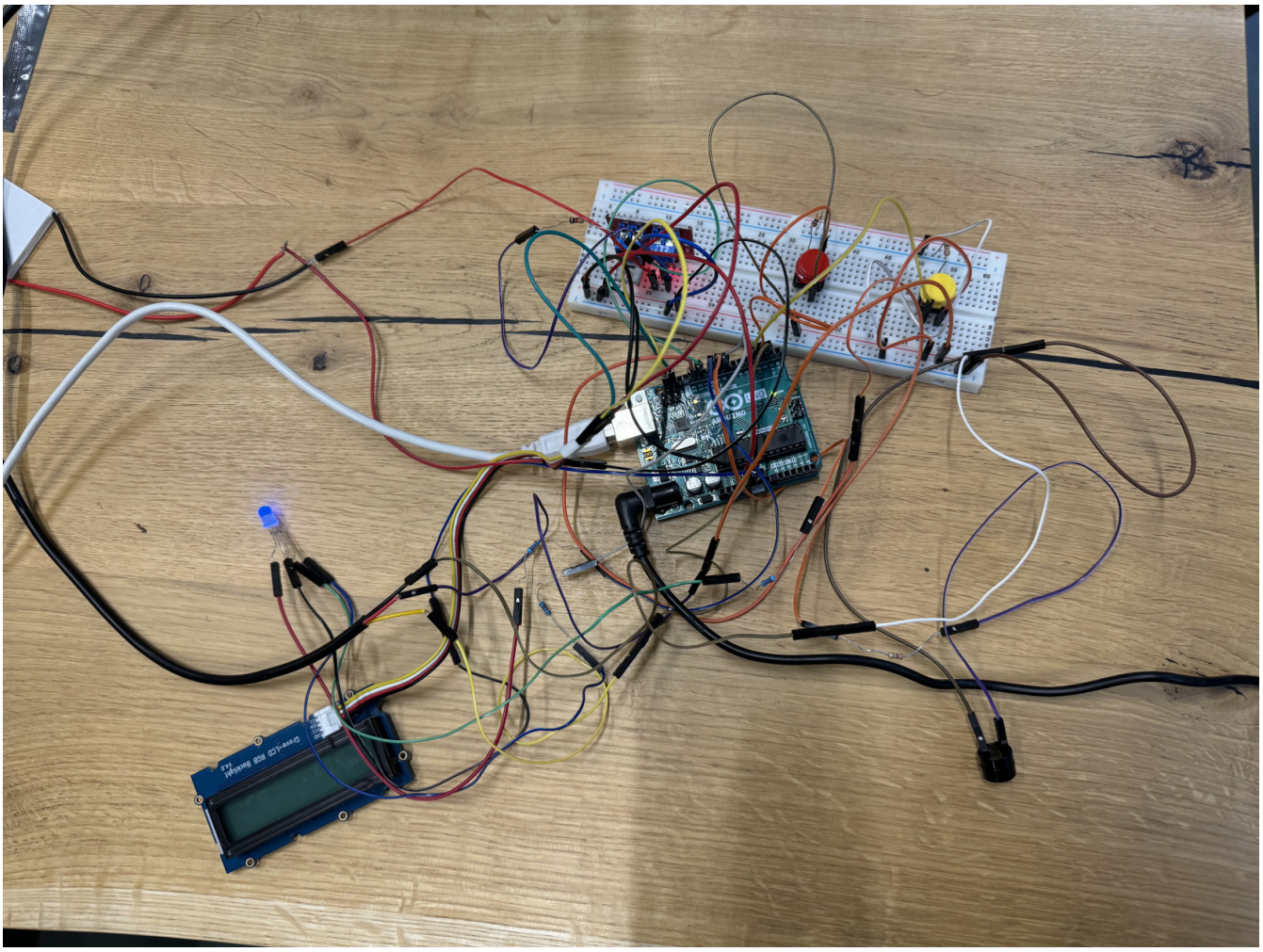


Figure 5 : Assemblage et câblage des composants

12/12/2024 :

- Deuxième réunion au Fablab.
- Plusieurs problèmes ont été rencontrés lors du montage et des tests.

> **Problèmes rencontrés pendant le montage : module Peltier, sonde de température, carte Arduino et écran LCD :(**

Problème avec le module Peltier : Le module Peltier ne chauffait pas. Pour vérifier si le problème venait du branchement, nous avons testé avec un moteur, et celui-ci a fonctionné correctement. Nous avons conclus que le module Peltier était défectueux et l'avons remplacé par un autre.

Problème avec la sonde de température : L'écran LCD affichait des valeurs de température instables. Nous avons trempé la sonde dans de l'eau chaude pour tester son fonctionnement, mais sans succès. La sonde semblait donc défectueuse et un nouvel essai avec une sonde différente a été prévu.

Problème avec la carte Arduino : La carte Arduino initiale ne fonctionnait pas correctement. Nous avons pris la décision de la remplacer par une autre, ce qui a résolu le problème et permis de continuer les tests.

Problème avec l'écran LCL : Le premier écran LCD était défectueux et a également été remplacé.

- Malgré ces remplacements, les valeurs affichées par la sonde restaient instables. Nous avons consulté notre tuteur, qui nous a conseillé d'ajouter une résistance de 4,7 Ω comme montré dans la photo ci-dessous.

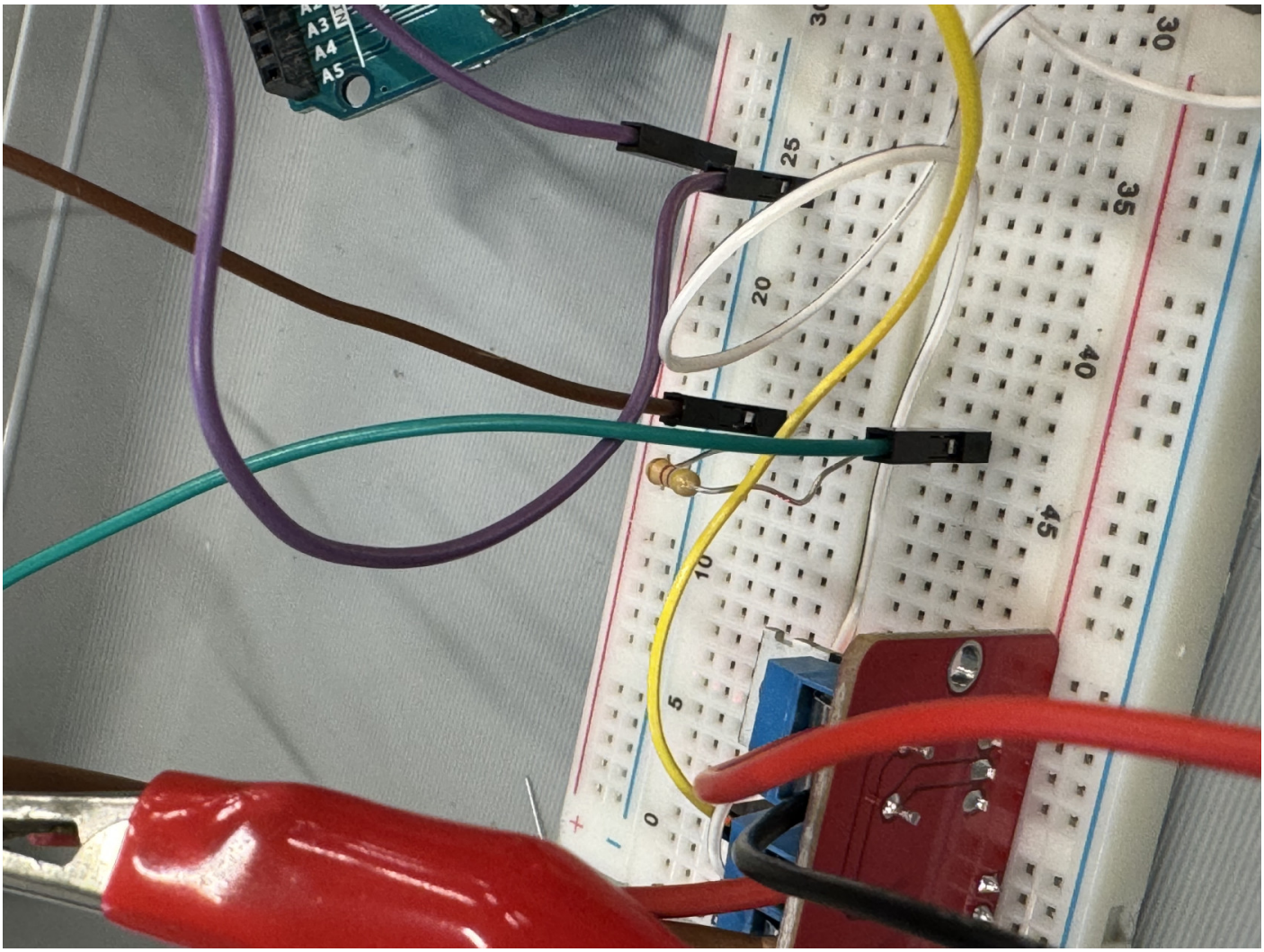


Figure 6 : Stabilisation des mesures de la sonde avec une résistance de 4,7 Ω

- Même après l'ajout de la résistance, la température ne se stabilise pas.
- Nous avons envisagé deux hypothèses possibles pour expliquer ce dysfonctionnement :

1. **La sonde pourrait être défectueuse**
2. **Problème de compatibilité avec la bibliothèque Arduino utilisée** : Il se pourrait qu'elle ne soit pas entièrement adaptée, ce qui pourrait entraîner des erreurs dans la lecture des données, nécessitant un ajustement du code pour résoudre ce problème.

- Afin de valider ces hypothèses, nous avons effectué une nouvelle tentative en utilisant un capteur différent. Le problème persistait.
- **Prochaine étape** : Révision approfondie du code pour identifier et résoudre le problème.

05/01/2024 :

- La connexion du capteur de température a été vérifiée, la résistance de 4,7 Ω devait être pontée entre le câble rouge 5 V et le câble jaune analogique. Une fois cela fait, le capteur est ensuite connecté au côté connexion numérique de la carte Arduino, qui dans le code révisé 2.0 est « *ONE_WIRE_BUS 7* » ou connexion numérique 7 en utilisant la bibliothèque *<OneWire.h>.*

- **Révision du code :**

Le code a été révisé et modifié pour intégrer la fonctionnalité PID permettant un contrôle total des valeurs constantes. Cela signifie qu'il peut désormais être ajusté pour obtenir la meilleure réponse en fonction de nos besoins. Dans ce cas, la constante de proportionnalité a été modifiée pour avoir une augmentation rapide de la température, ce qui entraîne inévitablement un dépassement. Donc, pour garantir une performance optimale, les constantes intégrale et différentielle doivent être modifiées en conséquence.
PID Controller V2.0 permet également l'activation ou la désactivation de la pompe de mélange tout en réduisant le décalage d'entrée entre toute pression sur un bouton et le temps de réponse de l'écran.
Enfin dans cette révision, l'instruction anti-rebond a été définie pour éviter toute pression accidentelle sur un bouton.

> **Le programme a été entièrement révisé et optimisé, et fonctionne désormais parfaitement, assurant ainsi une régulation de température précise et fiable pour le système :)**

Ainsi voici le programme qui fonctionne :

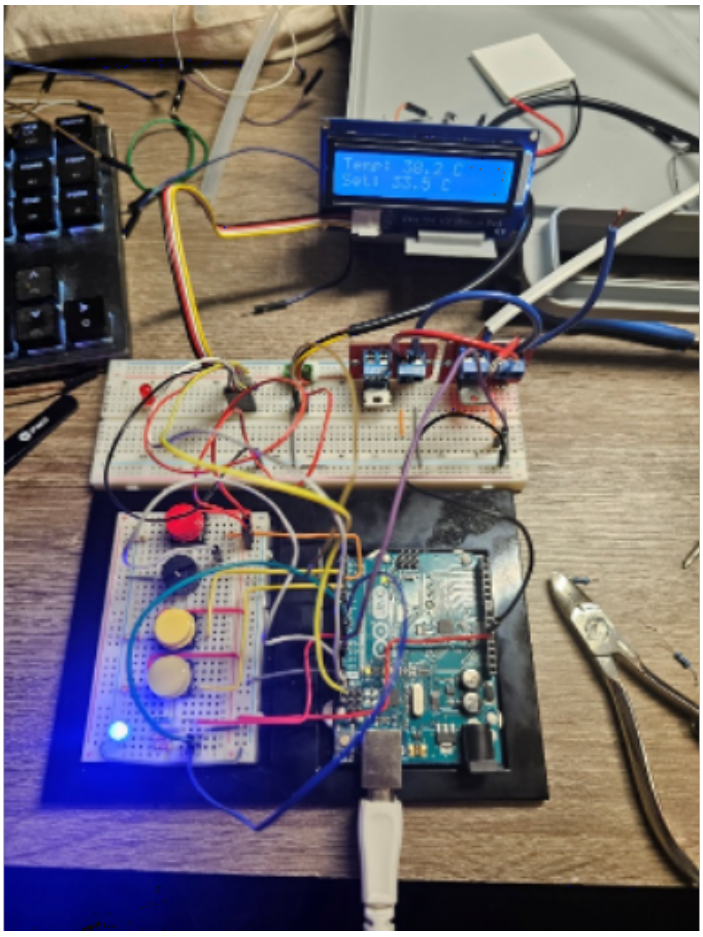
```
// PID COntroller V2.0
// Program has PID functionality, Pump on/off

#include <DallasTemperature.h>
#include <OneWire.h>
#include <Wire.h>
#include <rgb_lcd.h>
#include <PID_v1_bc.h>

// Pin Definitions
#define ONE_WIRE_BUS 7      // Pin for temperature sensor
#define BTN_UP_PIN 2        // Button to increase target temperature
#define BTN_DOWN_PIN 4      // Button to decrease target temperature
#define LED_R_PIN 10        // Pin for red LED
#define LED_G_PIN 9         // Pin for green LED
#define LED_B_PIN 8         // Pin for blue LED
#define MOTOR_PIN 12        // Pin to control the motor
#define PUMP_PIN 11         // Pin to control the pump

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

rgb_lcd lcd;

// PID Variables
double currentTemp = 0.0;     // Current temperature reading
double targetTemp = 30.0;     // Initial target temperature
double motorOutput = 0.0;     // PID output to control the motor
double Kp = 2.0, Ki = 0.5, Kd = 1.0; // PID constants

PID pidController(&currentTemp, &motorOutput, &targetTemp, Kp, Ki, Kd, DIRECT);
```

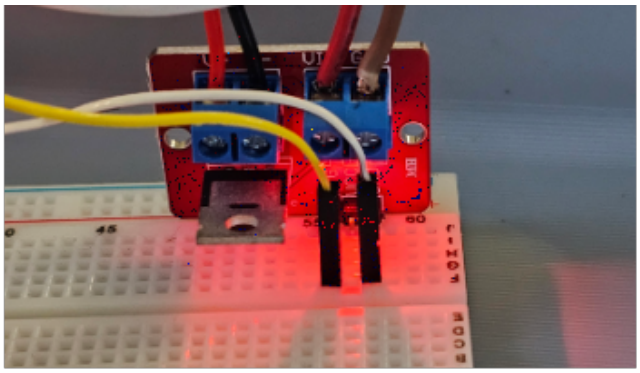
```
// Button Timing Variables
unsigned long lastButtonUpTime = 0;
unsigned long lastButtonDownTime = 0;
const unsigned long buttonPressInterval = 25; // Faster response time for buttons

// Pump Control Variables
bool pumpState = false;       // Tracks if the pump is ON or OFF
bool simultaneousPressDetected = false; // Tracks simultaneous press detection

void setup() {
 // Initialize LCD
 lcd.begin(16, 2);
 lcd.setRGB(255, 255, 255); // White backlight

 // Welcome message
 lcd.setCursor(0, 0);
 lcd.print("Regulation Temp");
 delay(2000);
 lcd.clear();

 // Pin configuration
 pinMode(BTN_UP_PIN, INPUT_PULLUP);
 pinMode(BTN_DOWN_PIN, INPUT_PULLUP);
 pinMode(LED_R_PIN, OUTPUT);
 pinMode(LED_G_PIN, OUTPUT);
 pinMode(LED_B_PIN, OUTPUT);
 pinMode(MOTOR_PIN, OUTPUT);
 pinMode(PUMP_PIN, OUTPUT);

 // Initialize LEDs and Pump
 digitalWrite(LED_R_PIN, LOW);
 digitalWrite(LED_G_PIN, LOW);
 digitalWrite(LED_B_PIN, LOW);
 digitalWrite(PUMP_PIN, LOW);

 // Start temperature sensor
 sensors.begin();
 Serial.begin(9600);

 // Initialize PID controller
 pidController.SetMode(AUTOMATIC);    // Set PID mode to automatic
```

```
void loop() {
  // Request temperature from sensor
  sensors.requestTemperatures();
  currentTemp = sensors.getTempCByIndex(0);

  // Display current temperature on LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(currentTemp, 1); // Display with 1 decimal place
  lcd.print(" C    ");     // Clear extra characters

  // Adjust target temperature with independent debounce logic
  if (digitalRead(BTN_UP_PIN) == LOW && millis() - lastButtonUpTime > buttonPressInterval) {
    targetTemp += 0.5;      // Increase target temperature
    lastButtonUpTime = millis();
  }
  if (digitalRead(BTN_DOWN_PIN) == LOW && millis() - lastButtonDownTime > buttonPressInterval) {
    targetTemp -= 0.5;      // Decrease target temperature
    lastButtonDownTime = millis();
  }

  // Display target temperature on LCD
  lcd.setCursor(0, 1);
  lcd.print("Set: ");
  lcd.print(targetTemp, 1); // Display with 1 decimal place
  lcd.print(" C    ");

  // Pump control: Detect simultaneous press of BTN_UP_PIN and BTN_DOWN_PIN
  if (digitalRead(BTN_UP_PIN) == LOW && digitalRead(BTN_DOWN_PIN) == LOW) {
    simultaneousPressDetected = true; // Simultaneous press detected
  }

  if (digitalRead(BTN_UP_PIN) == HIGH && digitalRead(BTN_DOWN_PIN) == HIGH && simultaneousPressDetected) {
    pumpState = !pumpState; // Toggle pump state
    digitalWrite(PUMP_PIN, pumpState ? HIGH : LOW); // Turn pump ON/OFF
    simultaneousPressDetected = false; // Reset detection flag
    delay(200); // Debounce delay for pump toggle
  }

  // Run PID computation
  pidController.Compute();
```

```
// Control motor output based on PID output
 analogWrite(MOTOR_PIN, motorOutput);

 // Control LEDs based on temperature state
 const float hysteresis = 1.0; // Temperature tolerance range
 if (currentTemp < targetTemp - hysteresis) {
   digitalWrite(LED_R_PIN, HIGH); // Red: Heating needed
   digitalWrite(LED_G_PIN, LOW);
   digitalWrite(LED_B_PIN, LOW);
 } else if (currentTemp > targetTemp + hysteresis) {
   digitalWrite(LED_R_PIN, LOW);
   digitalWrite(LED_G_PIN, LOW);
   digitalWrite(LED_B_PIN, HIGH); // Blue: Too hot
 } else {
   digitalWrite(LED_R_PIN, LOW);
   digitalWrite(LED_G_PIN, HIGH); // Green: Target temperature reached
   digitalWrite(LED_B_PIN, LOW);
 }

 delay(25); // Stabilization delay
}
```

- **Optimisation des connexions et amélioration de l'ergonomie**
  Dans la nouvelle révision du matériel, les connexions ont été réduites et réordonnées pour éviter les débranchements accidentels. Cette refonte simplifie également le transport du projet et améliore l'expérience de l'utilisateur en rendant le système plus accessible et facile à manipuler.





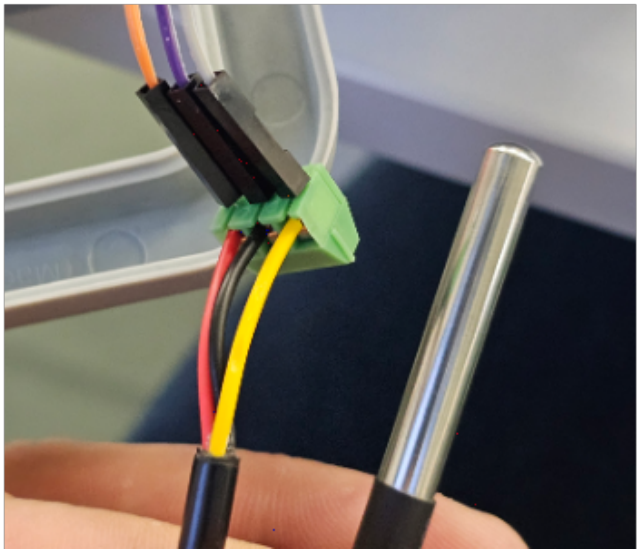


Figure 7 : Refonte du matériel : Simplification des connexions et amélioration de l'accessibilité

- **Evolutions futures**
  De nouvelles fonctionnalités sont en cours d'élaboration, tant sur le plan logiciel que matériel. Parmi les améliorations prévues, nous avons l'ajout de boutons de déclenchement supplémentaires permettant une éventuelle modification de la constante PID en temps réel. Une autre amélioration envisagée serait l'ajout d'un ventilateur de refroidissement pour accélérer le processus de refroidissement de l'eau chauffée en cas de besoin. Enfin, il est prévu d'utiliser le côté refroidissement du module Peltier pour refroidir un dissipateur thermique à eau connecté, afin de faciliter un refroidissement plus rapide de l'eau.

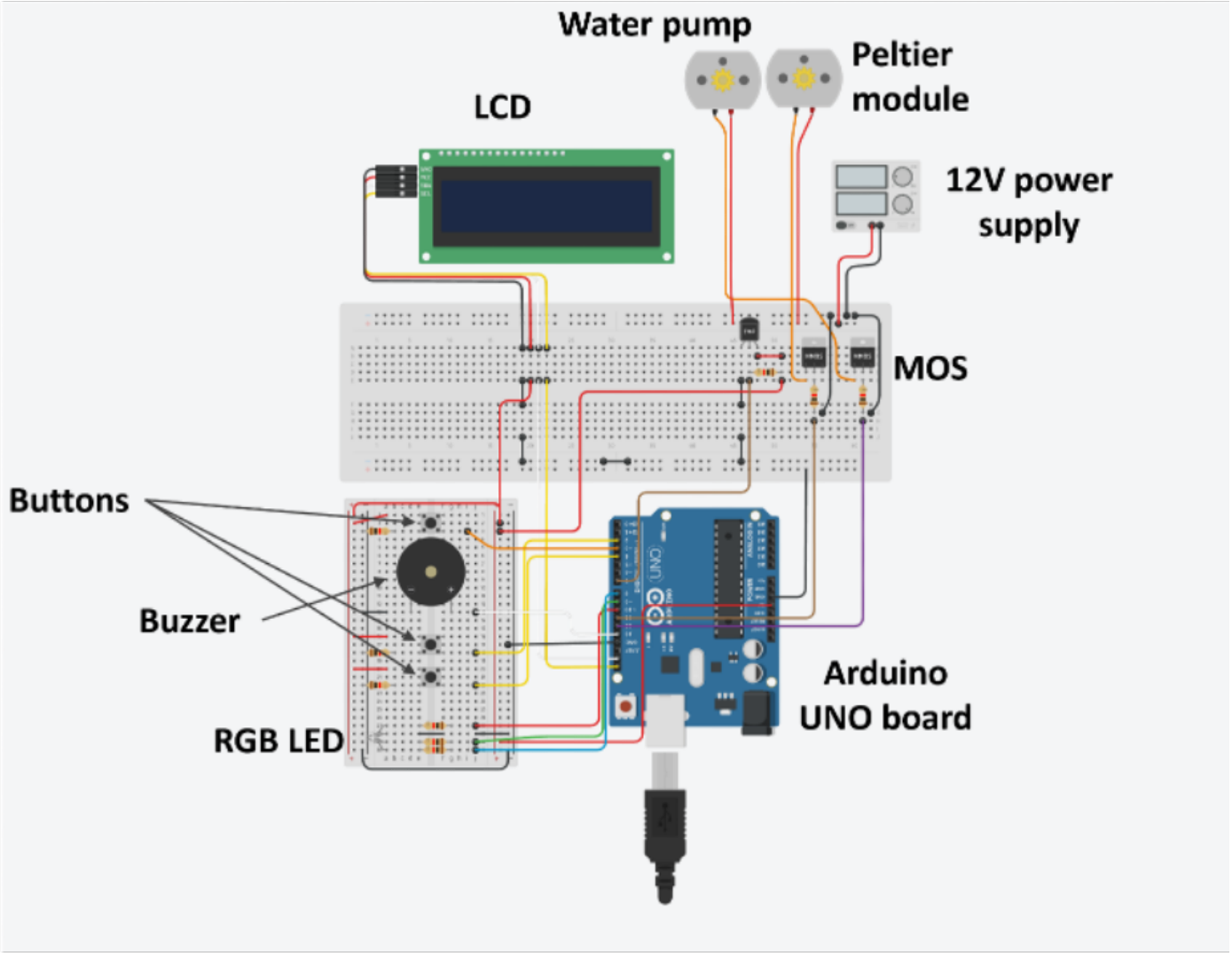


Figure 8 : Nouvelle simulation du système sur Tinkercad avec les nouvelles fonctionnalités

Notre système de régulation fonctionne désormais de manière fluide et précise assurant un contrôle optimal de la température comme prévu :)

Voici une vidéo illustrant les résultats obtenus après les dernières modifications. Cette démonstration met en avant l'efficacité du système mis en place.

24/01/2024 :

> **Problème rencontré pendant le test : la pompe ne fonctionne plus :(**

- Au cours de nos différents tests, la pompe a cessé de fonctionner, ce qui nous a conduits à en récupérer une nouvelle. À ce jour, le seul problème identifié reste celui de la pompe.
- Nous avons fixé un rendez-vous le lundi 27 janvier afin de présenter notre expérience.
- Par ailleurs, nous avons décidé d'intégrer un QR code à notre projet, permettant aux futurs étudiants d'accéder facilement à notre page Wiki.

27/01/2024 :

> **Problème rencontré pendant la finalisation du projet : le module de Peltier ne fonctionne plus :(**

- Récupération du Module Peltier car lors de la finalisation de notre projet, nous avons rencontré un imprévu le matin même : notre module Peltier initial a cessé de fonctionner. Afin de garantir le bon déroulement de la présentation, nous avons rapidement récupéré un nouveau module de remplacement, identique au précédent, et l'avons intégré au montage final.
- Impression au laser du QR code qui sera fixé sur le support contenant les informations clés du projet : nom, prénom, titre du projet et autres détails pertinents. Cette finition apportera une touche professionnelle et facilitera l'accès rapide à des ressources complémentaires ou à la documentation du projet via le QR code.
- Montage complet des composants sur le support rigide au Fablab, permettant une fixation sécurisée et une présentation soignée. Ce support intégrera tous les éléments nécessaires, tels que la carte Arduino, le module Peltier, le capteur de température, l'écran LCD et les boutons poussoirs, organisés de manière fonctionnelle.

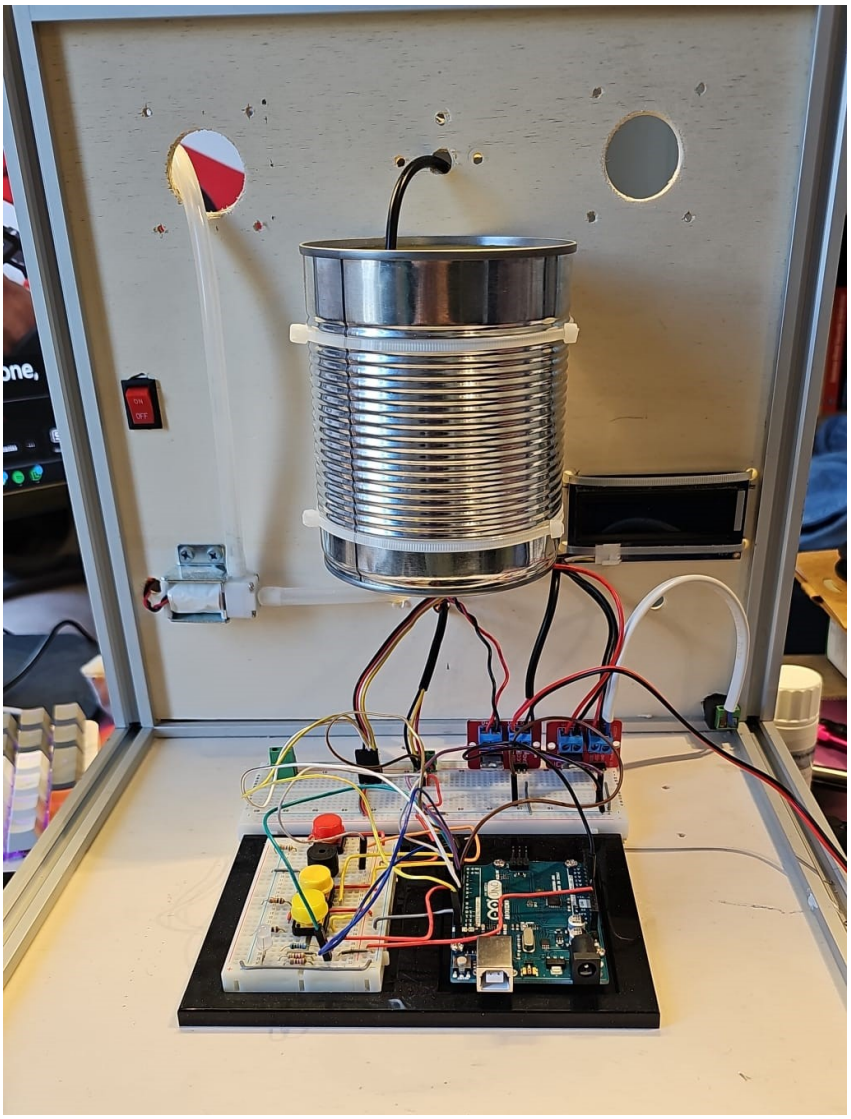


Figure 9 : Montage des composants sur le support

- Montage final de notre projet avant la présentation.

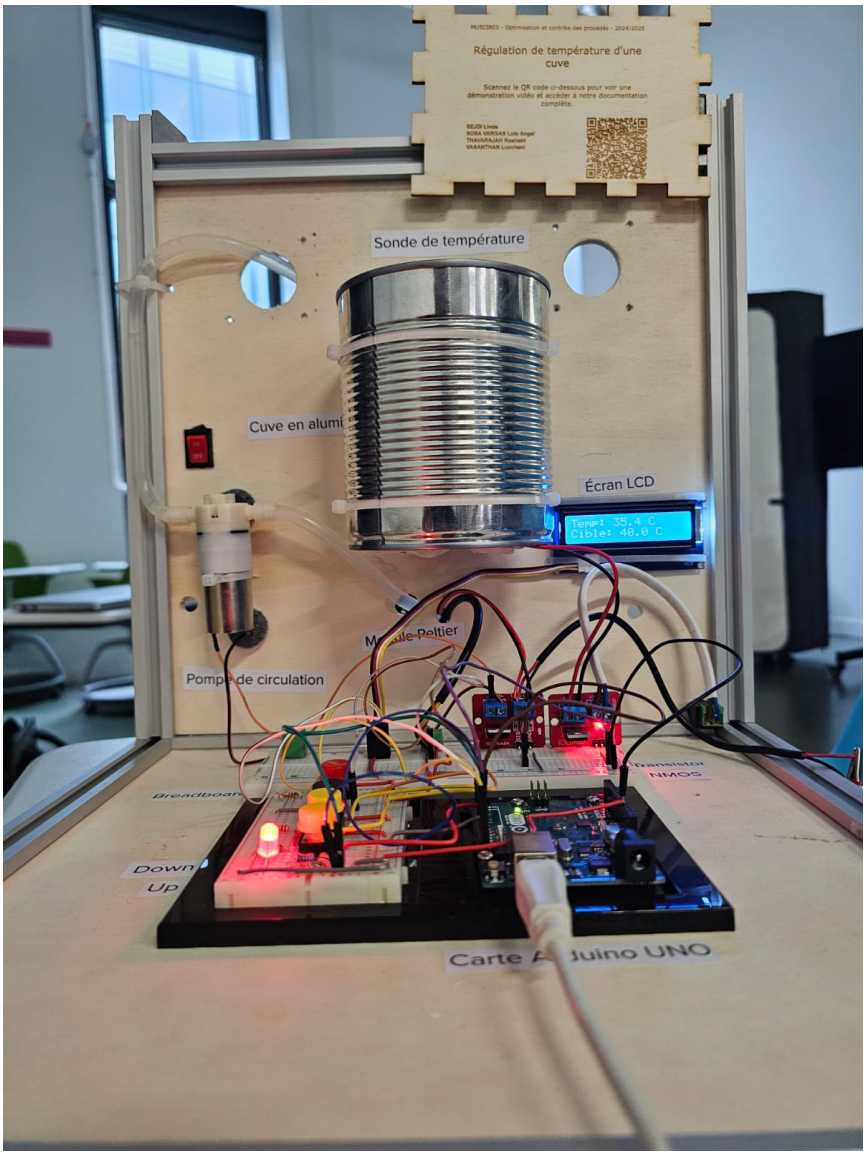


Figure 10 : Montage final de notre projet de régulation de la température d'une cuve

Dans cette vidéo, nous démontrons le fonctionnement du système de régulation de température à l'aide d'un module Peltier et d'une LED RGB. Nous expliquons comment la couleur de la LED change en fonction de la température mesurée par la sonde, offrant ainsi une visualisation intuitive du processus de régulation.

- **Rouge** : La LED passe au rouge lorsque la température mesurée est inférieure à la température cible. Cela indique que le système doit chauffer pour atteindre la consigne.
- **Vert** : La LED devient verte lorsque la température est stabilisée à la consigne, ce qui signifie que le système maintient la température idéale.
- **Bleu** : Si la température dépasse la consigne, la LED devient bleue, signalant que le système doit refroidir pour ajuster la température.

Cette fonctionnalité permet de suivre en temps réel l’état du système et garantit une régulation thermique précise et visuellement compréhensible. Dans la vidéo, nous mettons en évidence ces transitions de couleur en fonction des ajustements de la température cible, illustrant ainsi l’efficacité du système mis en place.

Conclusion :

Ce projet nous a permis d'appliquer nos connaissances théoriques en les transformant en une réalisation pratique fonctionnelle. Malgré les défis rencontrés, comme le remplacement du module Peltier en dernière minute, nous avons su faire preuve d'adaptabilité et de rigueur. Le montage final répond aux objectifs fixés, permettant un contrôle efficace de la température grâce à l'Arduino et aux différents composants intégrés. Ce projet nous a également permis de renforcer nos compétences en électronique, en programmation, et en gestion de projet. Il représente une étape importante dans notre apprentissage tout en ouvrant la voie à des améliorations futures.

> **Problèmes rencontrés dans le projet : le Module Peltier :**

Symptômes : Le module Peltier initialement utilisé ne chauffait pas correctement, perturbant ainsi la régulation thermique et affectant le processus global de contrôle de température. Après plusieurs tests, il a été conclu que le module était défectueux.

Solution : Le module défectueux a été remplacé par un modèle fonctionnel. Cependant, cette défaillance en fin de projet a ajouté des contraintes de temps, nécessitant une gestion rapide et efficace des ressources pour garantir la présentation du projet.

Le plus gros problème rencontré a été le module Peltier, que nous avons dû remplacer à plusieurs reprises. Au total, nous avons utilisé au moins cinq modules Peltier pour trouver un modèle fonctionnel et garantir la réussite de notre projet.

> **Améliorations possibles pour les années à suivre** :

- **Fiabilité des composants** :

Amélioration : Un contrôle plus rigoureux de la qualité des composants avant leur utilisation pourrait éviter des problèmes de défaillance, comme ceux rencontrés avec le module Peltier et la carte Arduino. De plus, un stockage approprié des composants et une gestion des stocks plus efficace pourraient minimiser les risques de panne en dernière minute.

- **Stabilité des mesures de température** :

Amélioration : Une investigation plus poussée sur la sonde de température et sur la bibliothèque utilisée pourrait permettre de résoudre définitivement les problèmes d'instabilité des lectures. Envisager l'utilisation de capteurs de température plus précis ou d'autres bibliothèques mieux adaptées à la régulation pourrait améliorer les performances du système.

- **Optimisation de l’interface utilisateur** :

Amélioration : L’expérience utilisateur pourrait être améliorée en simplifiant l’interface. Par exemple, l’ajout d’un écran tactile pour modifier la température cible directement ou l’utilisation d’un contrôle plus intuitif avec des boutons plus réactifs permettrait de mieux répondre aux attentes d’un utilisateur.

- **Extension des fonctionnalités logicielles** :

Amélioration : L’intégration de plus de fonctionnalités logicielles, comme la possibilité de visualiser l’historique des températures ou d’exporter les données de régulation sur un appareil externe (comme un smartphone ou un PC) via Bluetooth ou Wi-Fi, offrirait des capacités avancées pour les utilisateurs.

Revision #43
Created 10 October 2024 14:21:12 by Vasanthan Luxcheni
Updated 31 January 2025 11:23:12 by Thavarajah Roshani