Régulation de température d'une cuve

Informations :

Noms et prénoms :

SEJDI Linda (linda.sejdi@etu.sorbonne-universite.fr)
SOSA VERGAS Luis Angel (la.sosa.vergas@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 :

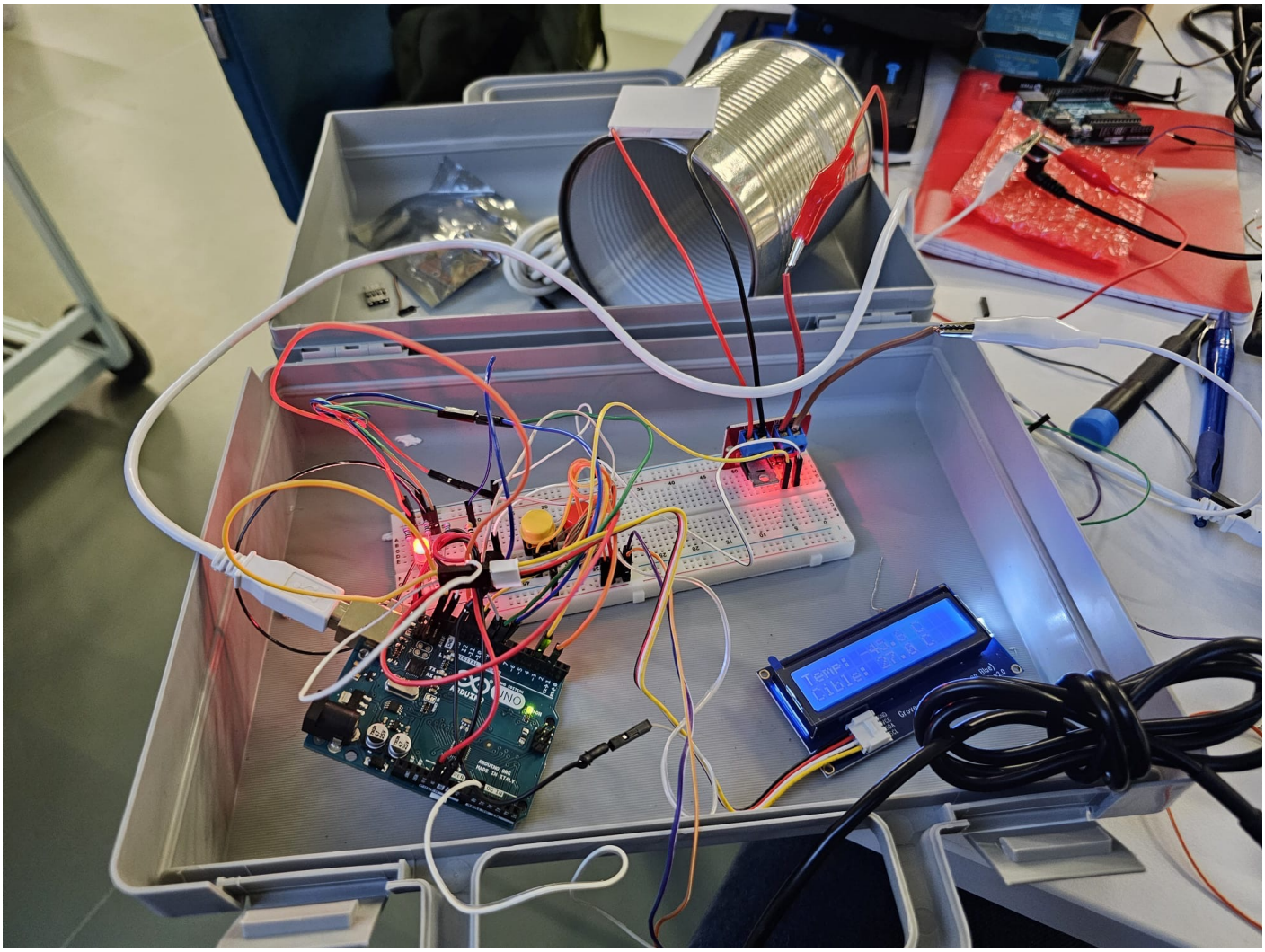
1. **Imposer une température cible :**
   La température cible est fixée dans le code Arduino au démarrage. Cette consigne initiale sert de référence pour la régulation.

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.



Matériel :

- Cuve en Aluminium

- Pompe de circulation
- Support pour la cuve
- Pince en C
- Module Peltier
- Pâte thermique
- Sonde DS1 (capteur de température)
- Transistor NMOS (IRF540N ou similaire)
- Alimentation externe de 12V (adapteur secteur ou batterie)
- Carte Arduino UNO
- Câble USB pour Arduino
- LED RGB
- Ecran LCD 16x2
- Buzzer
- Boutons poussoirs
- Platine d'essai
- Fils de connexion (10 mâle-mâle, 10 mâle-femelle, 10 femelle-femelle)
- Multimètre
- Fer à souder et étain
- Résistances (100 Ω, 220 Ω, 4,7 kΩ et 10 kΩ)

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 :

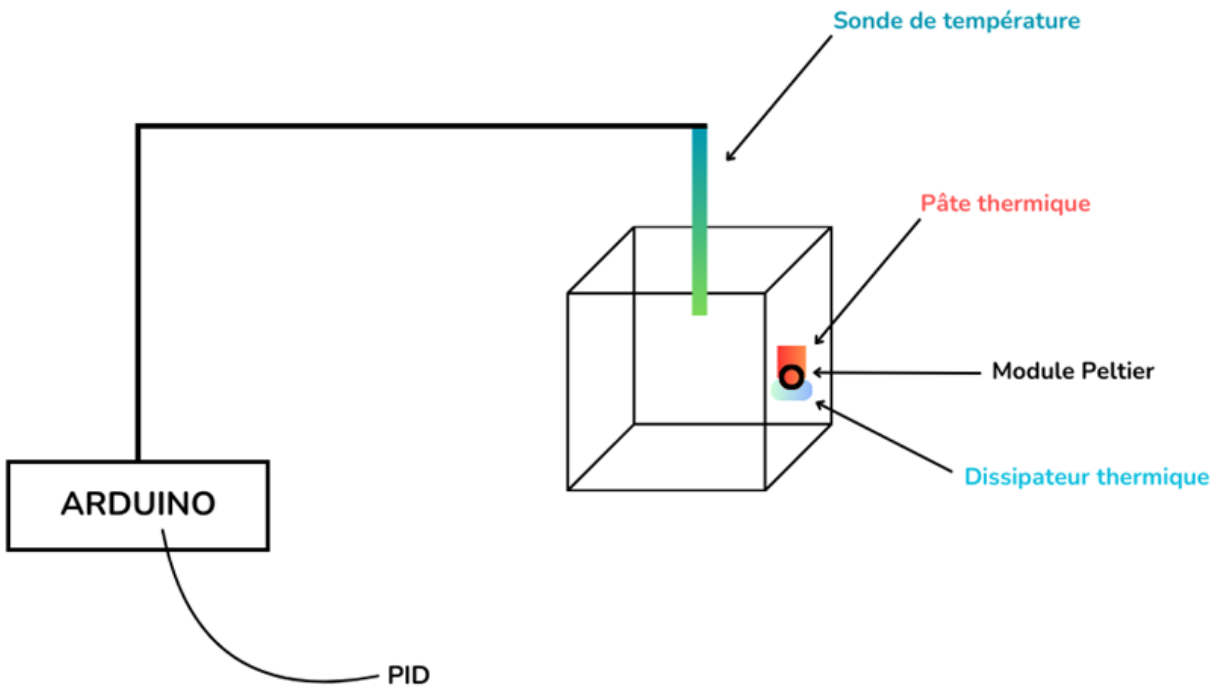


Figure 1 : Schéma du système de régulation de température avec Arduino et module Peltier

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.
- 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 validation de la liste.
- Récupération du matériel.
- Commande du module de Peltier (prévue pour mi-novembre).
- Prochaine étape : Simulation sur Tinkercad.

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.

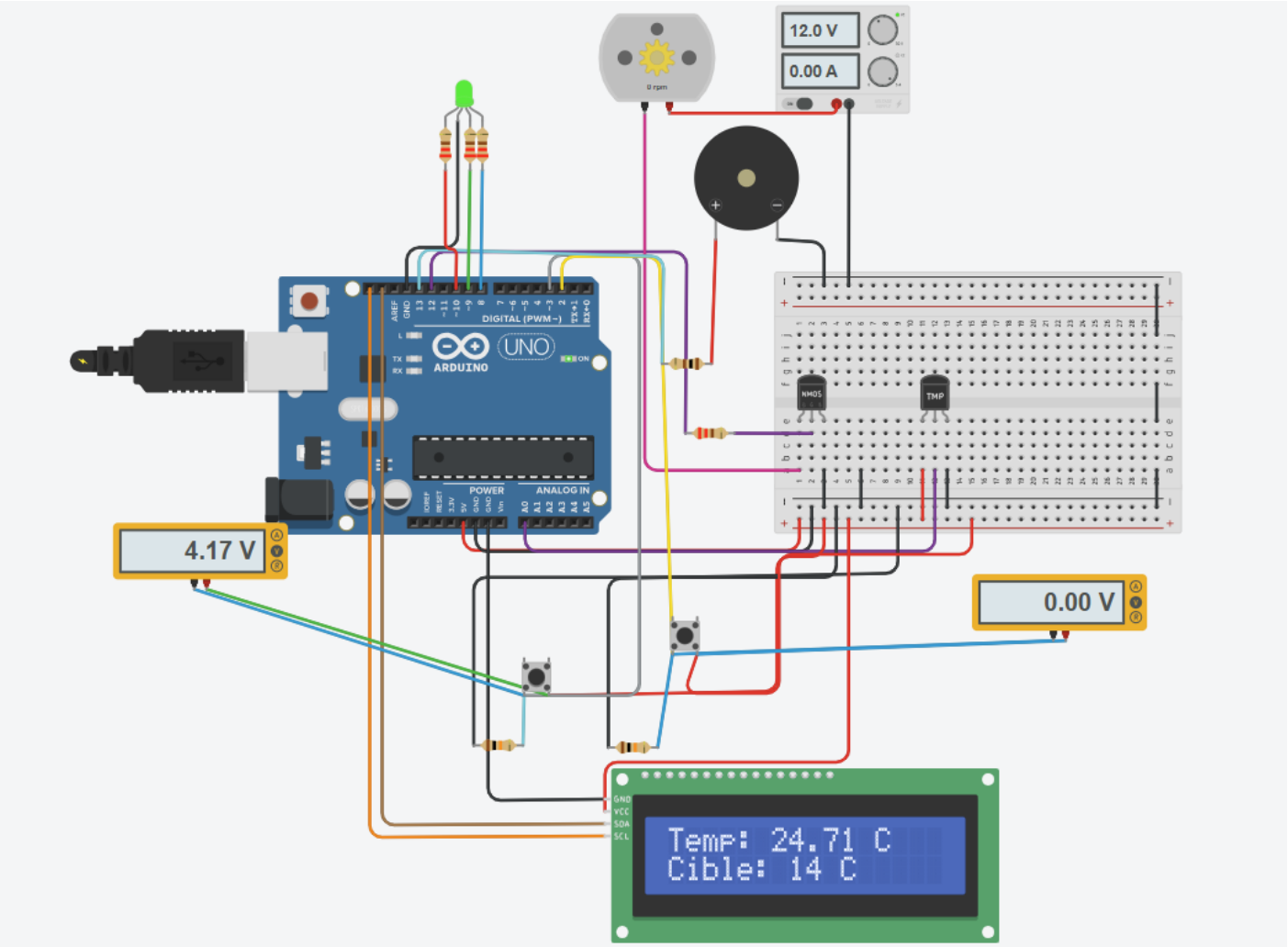
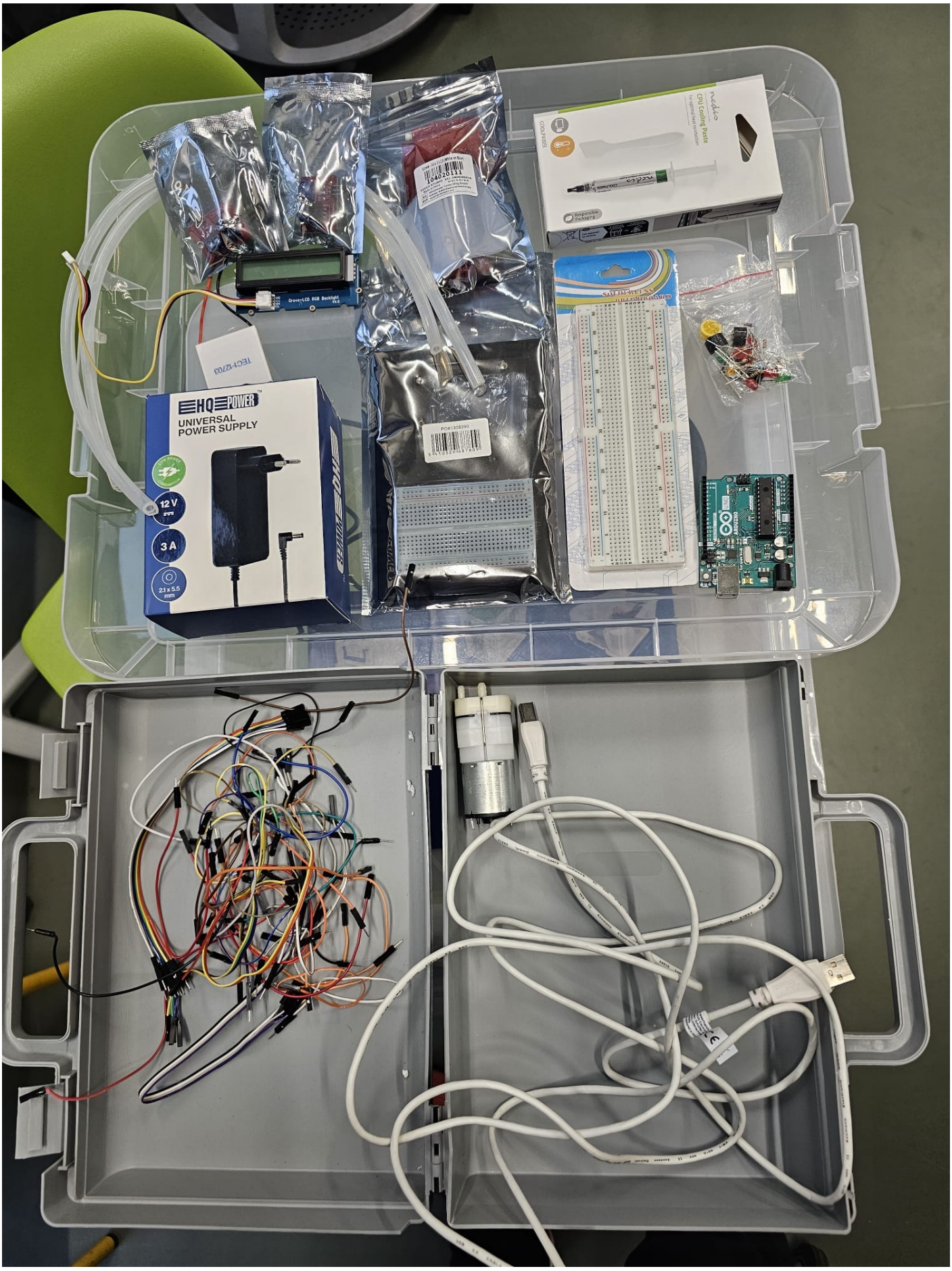


Figure 2 : Simulation du système sur Tinkercad

07/11/2024 :

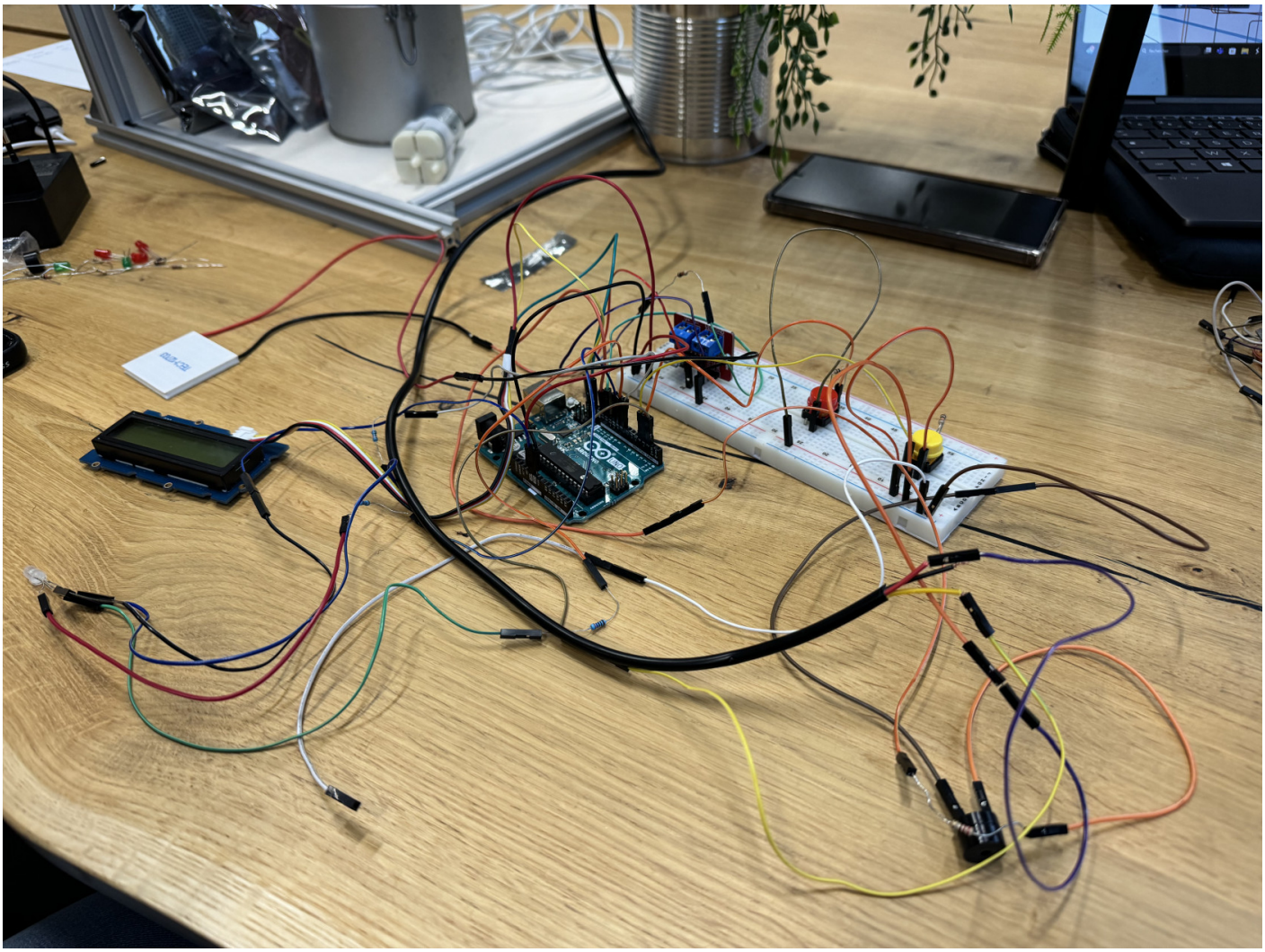
Récupération du matériel manquant :

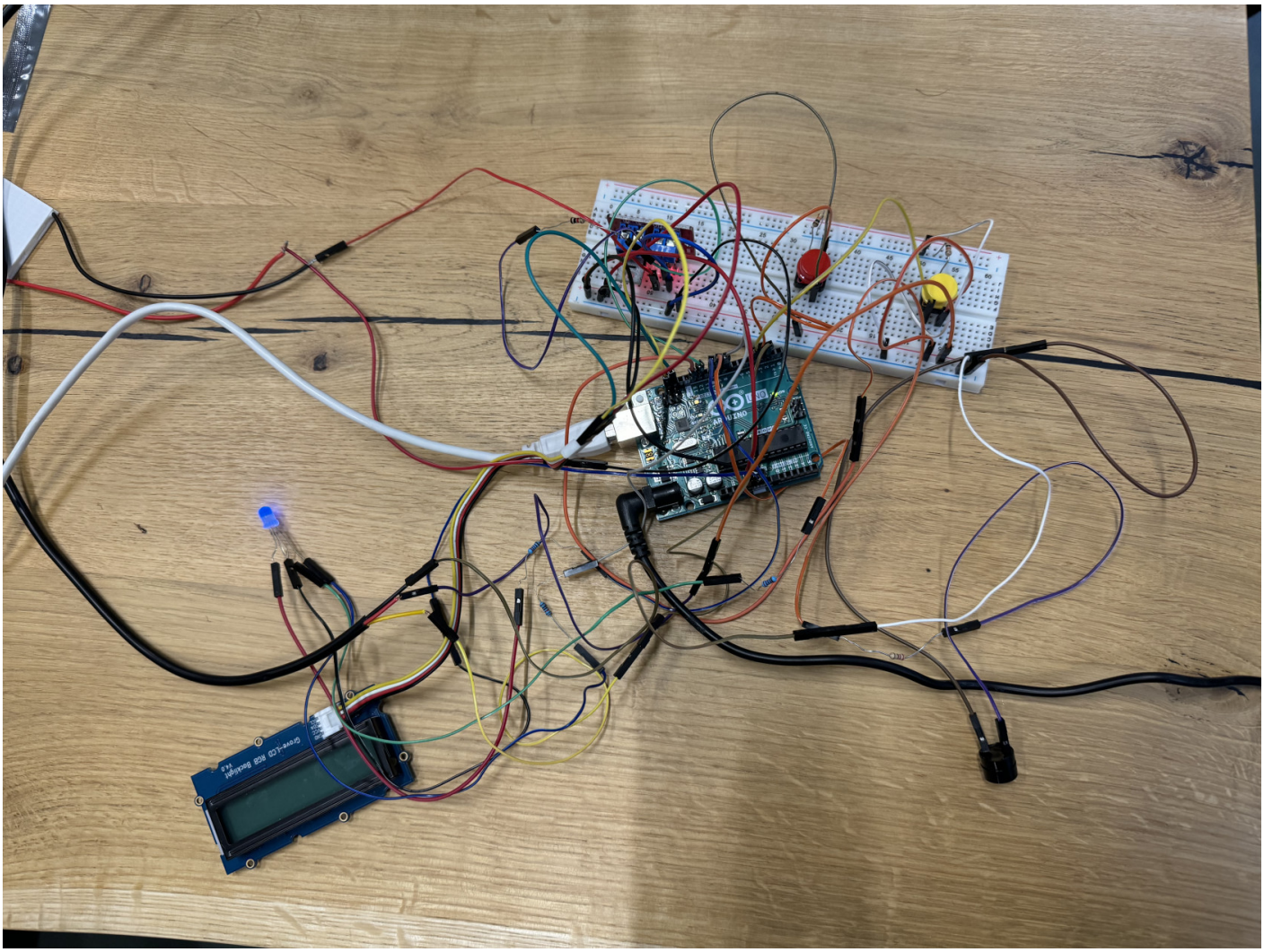
- Module Peltier
- Transistor MOS
- Breadboard
- Pâte thermique
- Ecran LCD


CPU Cooling Paste
HQ POWER
UNIVERSAL
POWER SUPPLY
12V
3A
21 x 5.5 mm

21/11/2024 :

- Première réunion au FabLab.
- Mise en place du montage.
- Problème de port identifié.

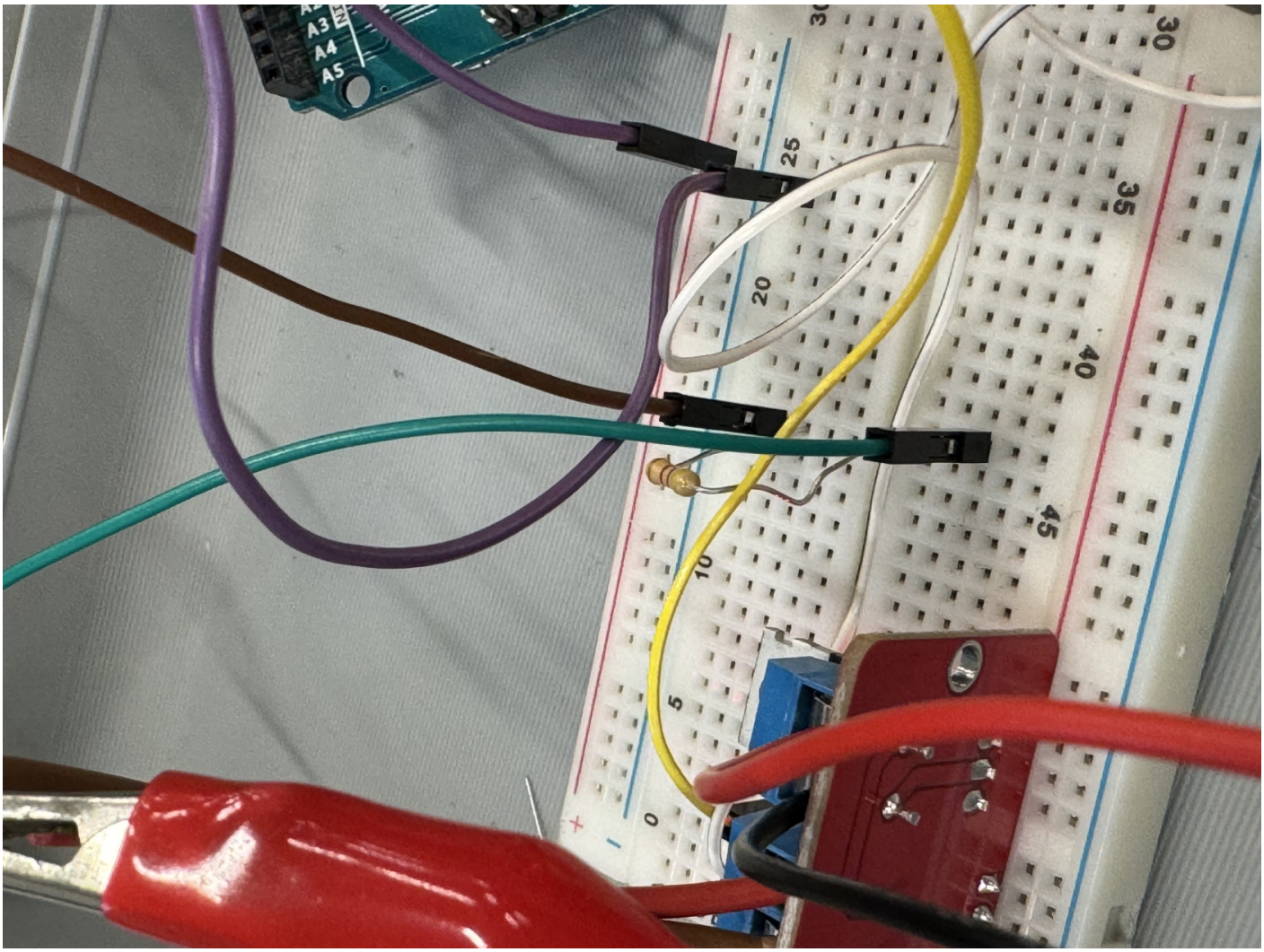




12/12/2024 :

- Deuxième réunion au Fablab.
- Problème avec le module Peltier : Le module Peltier ne chauffait pas. Pour vérifier le branchement, nous avons testé avec un moteur, qui a bien fonctionné. Nous avons conclus que le module Peltier était défectueux et l'avons remplacé par un autre.
- Problème avec la sonde de température : L'écran affichait des valeurs instables. Nous avons trempé la sonde dans de l'eau chaude pour tester son fonctionnement, mais sans succès.
- La carte Arduino initiale ne fonctionnait pas correctement. Nous l'avons remplacée.
- Le premier écran LCD était défectueux et a également été remplacé.

Malgré le remplacement du module Peltier, les valeurs éffichées par la sonde restaient instables. Nous avons consulté le tuteur, qui nous a conseillé d'ajouter une résistance de 4,7



Même avec la résistance, la température ne se stabilise pas. Nous avons envisagé deux hypothèses :

1. La sonde est peut-être défectueuse.
2. Il pourrait y avoir un problème de compatibilité avec la bibliothèque Arduino utilisée, nécessitant un ajustement du code.

- Nouvelle tentative avec un capteur différent. Le problème persiste.
- Prochaine étape : Révision approfondie du code pour identifier et résoudre le problème.

Revision #14
Created 10 October 2024 14:21:12 by Vasanthan Luxcheni
Updated 17 December 2024 15:04:21 by Thavarajah Roshani