Régulation de la température d'une réaction exothermique simulée

Régulation de la température d'une réaction exothermique simulée

L’objectif de notre projet est de réguler la température à l’intérieur d’un réacteur à 20°C par la mise en place d’un système automatisé qui va selon les conditions activer une pompe pour permettre au fluide caloporteur de circuler et ainsi abaisser la température dans le réacteur. L’eau présente à l’intérieur du réacteur monte en température via l'introduction dans le réacteur d'une poche de gel autochauffante. L’objectif est donc de simuler une réaction exothermique et de contrôler la température du réacteur afin d’éviter tout emballement thermique.

Noms et Prénoms :

- Halil Ayturk (halil.ayturk@etu.sorbonne-universite.fr)
- Taoufik Karrab (taoufaik.karrab@etu.sorbonne-universite.fr)
- Basile Linon (basile.linon-gervais@etu.sorbonne-universite.fr)
- Sofiane Meridji (Sofiane.meridji@etu.sorbonne-universite.fr)

Date de début : 01/09/2023

Date de fin : 02/02/2024

Nous avons dans un premier temps fait un grafcet de notre projet pour décrire les différentes étapes de notre système de régulation pour pouvoir ensuite faire une liste des différents composants qui vont être demandés.

Capteur de température numérique DS18B20 étanche	× 1
Pompe	× 2

Motor Shield V2.0	× 1
Arduino Uno	× 1
Résistance pull-up de 4,7 kΩ	× 1
Fils de liaison	~ 15
Résistance led	× 1
Câble USB type A/B	× 1
Alimentation de 12V	X 1
Serpentin métallique	X 1
Tuyaux	X 4

fig.1 Tableau des différents composants électroniques utilisés

Nous avons par la suite développé le système du capteur de température.

Nous avons tout d’abord simulé sur Tinkercad le montage du circuit pour voir si le code que nous avions implémenté était opérationnel.

Image not found or type unknown

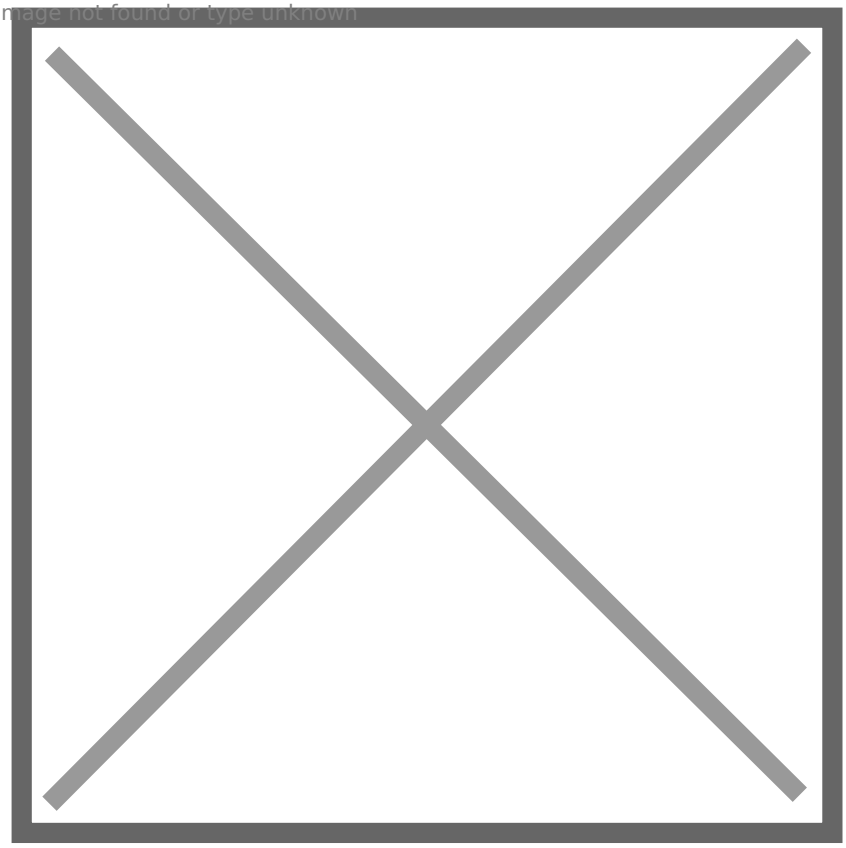

fig.2A Schéma de montage du capteur de température étanche sur la carte arduino

Fil noir (GND)	Terre
Fil jaune (DQ)	Pin 2 Arduino et via une résistance de 4,7 kΩ à 5V

Fil rouge (Vdd)	5V

fig.2B Tableau des connexions du capteur de température numérique

Une fois le code validé via Tinkercard, nous avons assemblé le capteur de température sur la carte Arduino en utilisant des fils de liaisons. L’assemblage a été réalisé par soudage en utilisant de l'étain. Après avoir monté le circuit électrique, l'objectif était de déterminer si le capteur de température était directement fonctionnel ou si il demandait une étape de calibrage.

Pour cela, nous avons implémenté un code qui avait pour rôle de mesurer la température du milieu et de l’afficher sur l'application Arduino. Le capteur de température DS18B20 intègrent des circuits de commande qui simplifient la programmation grâce à des bibliothèques préconfigurées, permettant ainsi de convertir le signal électrique en température. Un test qui consistait à mesurer la température du milieu à ensuite été réalisé (eau chaude, eau froide) en plongeant notre capteur dans un récipient avant de lire la température affichée puis de la comparer avec la valeur fournie par un thermomètre. Dans notre cas, le capteur de température était déjà calibré et prêt à l’emploi.

Il a ensuite fallu réussir à faire fonctionner la pompe avec une alimentation supplémentaire de 12V sans endommager la carte Arduino. Nous avons donc décidé d’utiliser un “motor shield” qui est un dispositif conçu pour contrôler et alimenter des moteurs. Il s'agit d'une carte d'extension que l'on place sur une carte Arduino pour faciliter le contrôle des moteurs. Les motor shields intègrent des circuits de commande de moteurs et simplifient la programmation grâce à des bibliothèques préconfigurées, permettant ainsi de contrôler facilement la vitesse, la direction, et d'autres aspects des moteurs. Tout comme le capteur de température, la liaison entre la pompe et la carte arduino nécessite l’utilisation de fils de liaison et l’assemblage a été réalisé par soudage. Après assemblage du motor shield sur la carte arduino et montage du circuit électrique, nous avons testé la pompe en implémentant un code qui avait pour but d’activer la pompe pendant 2s puis de l’arrêter 5s et cela de manière répétitive.

Image not found or type unknown

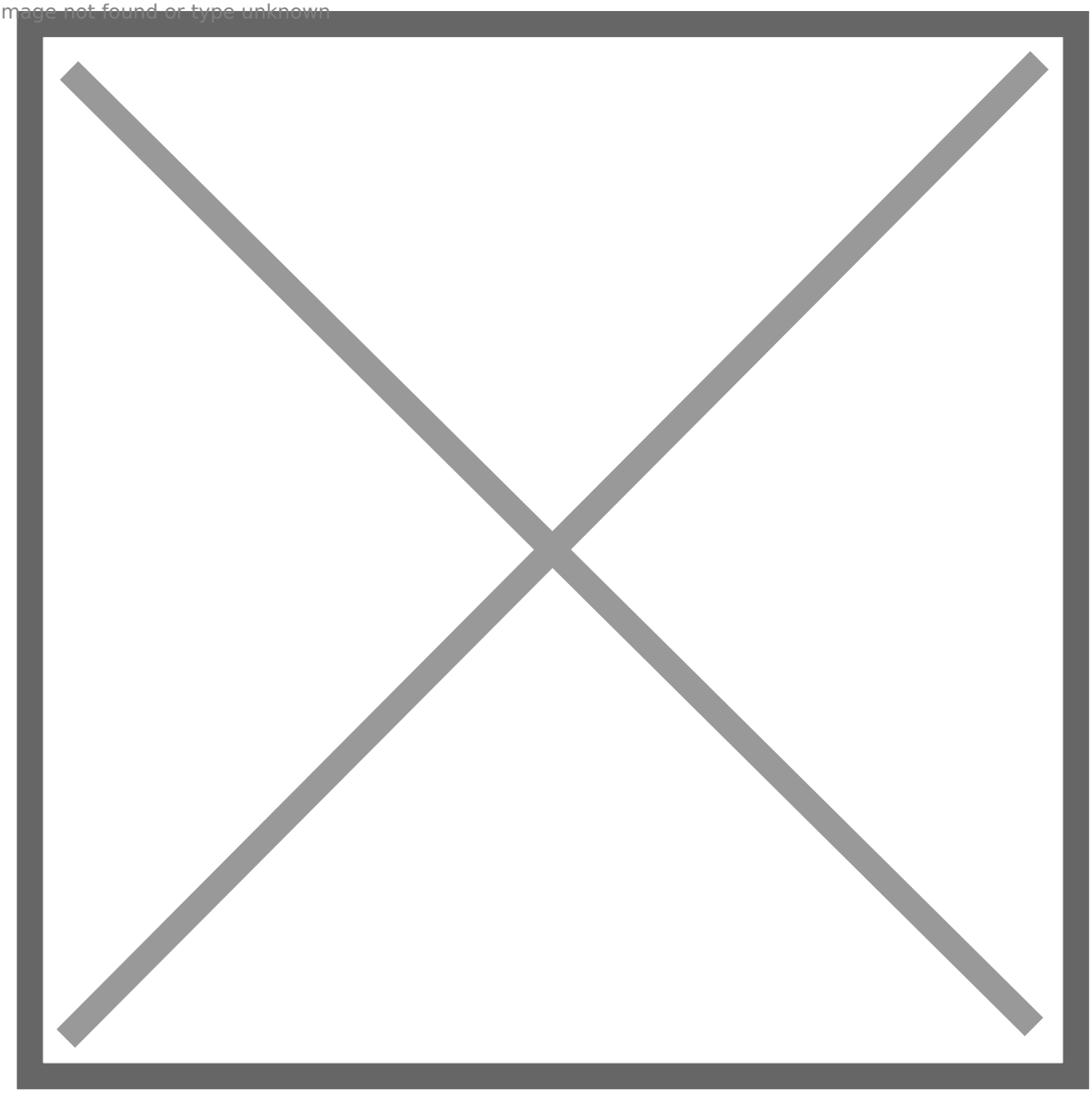

fig.3 Montage de la pompe sur le moteur 1 du shield motor

La troisième étape était de réaliser le circuit électrique qui est la combinaison des deux premières étapes. Nous avons ensuite fusionné les deux premiers code en ajoutant une condition sur la température. La carte arduino avait donc pour rôle de faire fonctionner la pompe si et seulement si la température du milieu était supérieure à 20°C. Nous avons ensuite fait un test qui consistait à mesurer la température du milieu (eau chaude) pour voir si la pompe s'actionne bien lorsque la température était bien au-dessus de 20°C.

Le code final est donc le suivant:

```
#include <Wire.h>

#include <Adafruit_MotorShield.h>
```

```
#include "utility/Adafruit_MS_PWMServoDriver.h"

#include <OneWire.h>

#include <DallasTemperature.h>



// Pin de données du capteur DS18B20

const int pinDonneesDS18B20 = 2;  // Remplacez le numéro de la broche par celui que vous utilisez



// Seuil de température pour activer la pompe (20°C)

const float seuilTemperature = 20.0;



// Création d'une instance OneWire pour la communication avec le capteur

OneWire oneWire(pinDonneesDS18B20);



// Création d'une instance DallasTemperature pour la gestion du capteur

DallasTemperature sensors(&oneWire);



// Initialisation du shield moteur

Adafruit_MotorShield AFMS = Adafruit_MotorShield();



// Création d'une instance du moteur 1

Adafruit_DCMotor *moteurPompe = AFMS.getMotor(1);



void setup() {
```

```
  // Initialisation du port série
  Serial.begin(9600);
  Serial.println("Adafruit Motorshield v2 - Pompe Activation");

  // Initialisation du capteur
  sensors.begin();

  // Initialisation du shield moteur
  AFMS.begin();
}

void loop() {
  // Demande au capteur de lire la température
  sensors.requestTemperatures();

  // Lecture de la température en degrés Celsius
  float temperatureCelsius = sensors.getTempCByIndex(0);

  // Vérification si la lecture est valide
  if (temperatureCelsius != DEVICE_DISCONNECTED_C) {
    // Affichage de la température sur le moniteur série
    Serial.print("Température : ");
    Serial.print(temperatureCelsius);
    Serial.println(" degrés Celsius");
```

```
    // Activation de la pompe si la température est supérieure au seuil
    if (temperatureCelsius > seuilTemperature) {
      moteurPompe->setSpeed(255);  // Vitesse maximale
      moteurPompe->run(FORWARD);   // Sens horaire
      Serial.println("Pompe activée !");
    } else {
      moteurPompe->setSpeed(0);    // Arrêt du moteur
      Serial.println("Pompe désactivée");
    }
  } else {
    Serial.println("Erreur de lecture du capteur");
  }

  // Attente d'une seconde avant la prochaine mesure
  delay(1000);
}
```

Dans l'ordre d'assemblage nous avons monté tout d’abord le support qui nous a été fourni afin de fixer les différents composants. Une fois le support monté, nous avons fixé les pompes, la carte Arduino et le breadboard sur le support en perçant des trous. Nous avons ensuite pris les dimensions pour pouvoir couper les différents tuyaux dans lesquels le fluide caloporteur (eau) va circuler. Comme la double enveloppe ne nous a pas été fournie pour des raisons financières, nous avons initialement placé un bocal (réacteur) dans un bocal plus grand qui fait office de double enveloppe. Ce montage s'est montré concluant lors de nos tests mais après discussion avec les techniciens de la plateforme IC un serpentin métallique dans lequel le fluide caloporteur peut se déplacer à été récupéré et monté dans une tasse métallique simulant alors un réacteur ouvert avec passage de fluide caloporteur dans le milieu réactionnel.

Image not found or type unknown

Revision #2
Created 30 January 2024 14:39:21 by Linon Basile
Updated 30 January 2024 15:28:05 by Linon Basile