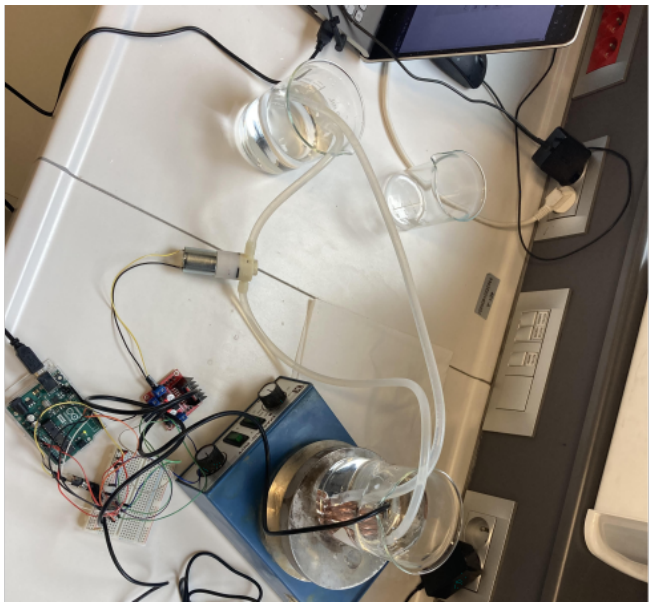
Régulation de température grâce à un microcontrôleur Arduino et à un pont en H L298N

Projet de régulation de température grâce à un microcontrôleur Arduino et un pont en H L298N réalisé dans le cadre de l'UE 5Ci803, année scolaire 2023-2024.

Cette régulation peut être transposée à l'échelle industrielle, pour réguler la température d'un milieu réactionnel afin de contrôler le risque d'emballement thermique d'une réaction exothermique par exemple.



Réalisateurs :

- Florent GALLERET
- Khaled SERHANE
- Christelle BERAUD
- Audrey Tessa TCHONLA TCHIBONSOU

Matériel :

- Pompe
- Pont en H L298N
- Sonde de température DS18B20
- Résistance 4.7 kOhm
- Microcontrôleur Arduino UNO
- Breadboard
- Alimentation 12V
- Fils électriques
- Tuyaux d'eau
- Bécher
- Plaque chauffante
- Eau fraîche
- Gaine thermique (cuivre)

Description du système

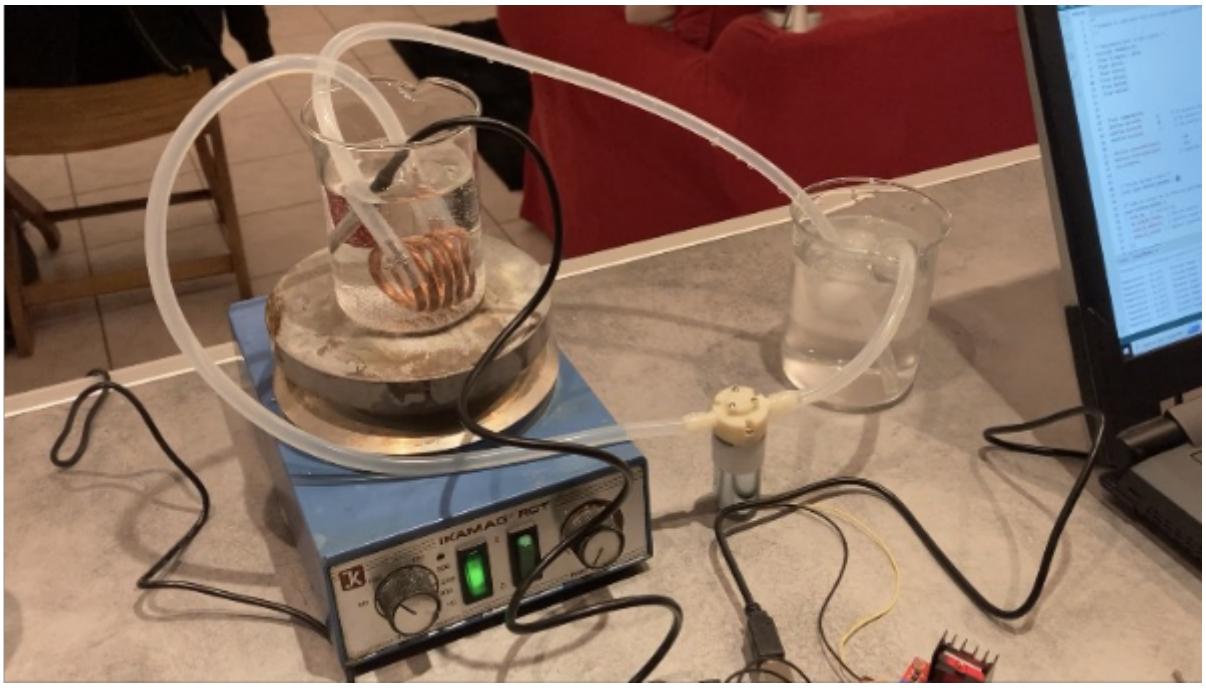
Le système a refroidir est un système matérialisé par de l'eau portée à différentes températures placée dans un bécher dans lequel est placé une gaine thermique en cuivre et chauffé par une plaque chauffante.

La pompe (ici symbolisée par Moteur) permet de pomper de l'eau fraîche d'un autre bécher, passant par les tuyaux d'eau et la gaine thermique.

Le pont en H L298N permet de faire tourner le moteur à différents régimes lorsqu'on est plus ou moins éloigné de la température de consigne. Ici, nous avons défini plusieurs valeurs de température pour lesquelles différentes vitesses de pompage sont définies. En effet, plus on s'approche de la température de régulation, plus le moteur va tourner à régime réduit, pour finalement se couper (voir étape 3). Cet élément alimente l'ensemble du système en 12V grâce à l'alimentation.

La sonde de température permet de mesurer la température à l'instant T dans le système.

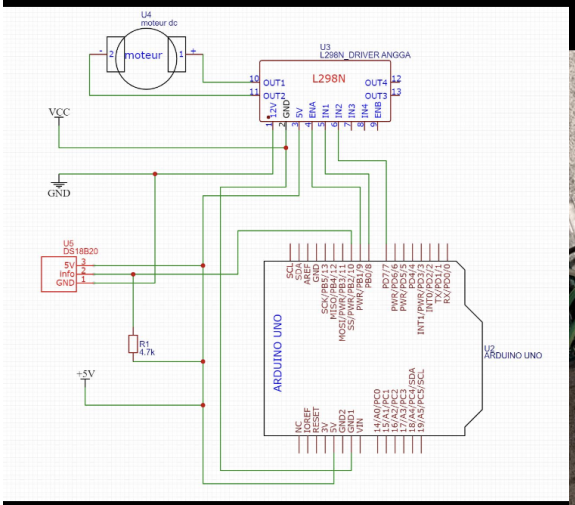
Le microcontrôleur Arduino permet de faire communiquer entre eux les éléments du montage et de relier l'ensemble des composants du système ensemble, par l'intermédiaire de la breadboard.

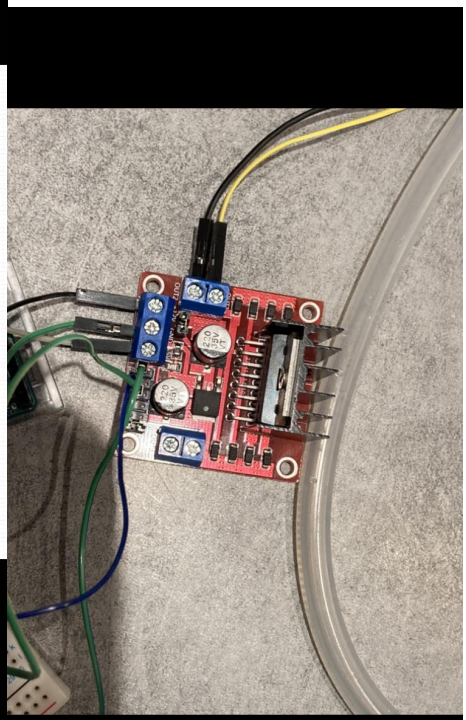


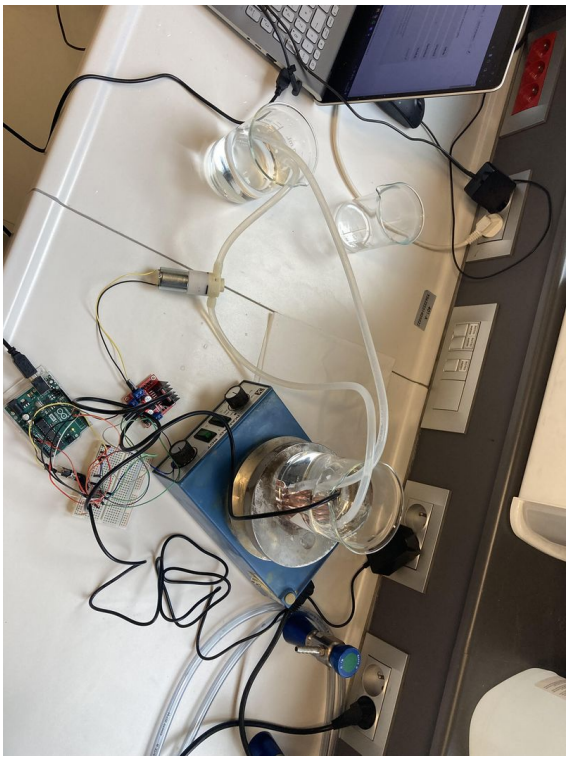
Connexion des éléments du système au microprocesseur Arduino

Pour mettre en œuvre le système, nous connectons différents éléments (pont en H, microcontrôleur Arduino, pompe, sonde de température, résistance) à la breadboard afin qu'ils

[redacted]espondant aux éléments connectés aux broches dans le







Développement du code pour microcontrôleur Arduino

C'est le coeur de la régulation.

Ce code permet de définir précisément la manière dont le système sera régulé.

Ici, une température de consigne (T_regule) est définie à 28°C ainsi que différentes autres températures (delta1, delta2, delta3, delta 4 et delta5). Elles sont définies plus tard dans la boucle de configuration (void_setup) respectivement pour T_regule-3, T_regule-2, T_regule-1, T_regule+1_ Tregule+2 afin que le moteur tourne à différents régimes (régime maximal pour une température à plus de 2 degrés de la température de consigne, puis une diminution progressive du régime à mesure que l'on se rapproche de la température de consigne) pour finalement se couper lorsqu'il atteint T_regule-2. L'utilisateur voit directement sur l'interface Arduino IDE la température à laquelle est le système à l'instant T et la vitesse du moteur.

```
/* Dépendance pour le bus 1-Wire */

#include <OneWire.h>

float T_regule = 28.0;

float delta1;
```

```
float delta2;

float delta3;

float delta4;

float delta5;

float Temperature;

#define borneENA 9 // On associe la borne "ENA" du L298N à la pin D9 de l'arduino

#define borneIN1 8 // On associe la borne "IN1" du L298N à la pin D8 de l'arduino

#define borneIN2 7 // On associe la borne "IN2" du L298N à la pin D7 de l'arduino

#define vitesseMinimale 60 // Rapport cylique minimal du signal PWM, pour faire tourner le moteur au minimum de sa vitesse (en pratique, on évitera de trop approcher la valeur 0)

#define vitesseMaximale 255 // Rapport cylique maximal du signal PWM, pour faire tourner le moteur au maximum de sa vitesse

int vitesse; // Spécifie la vitesse de rotation du moteur, entre son minimum (0) et son maximum (255) <= signal PWM

/* Broche du bus 1-Wire */

const byte BROCHE_ONEWIRE = 10;

/* Code de retour de la fonction getTemperature() */

enum DS18B20_RCODES {

READ_OK, // Lecture ok

NO_SENSOR_FOUND, // Pas de capteur

INVALID_ADDRESS, // Adresse reçue invalide

INVALID_SENSOR // Capteur invalide (pas un DS18B20)

};

/* Création de l'objet OneWire pour manipuler le bus 1-Wire */

OneWire ds(BROCHE_ONEWIRE);
```

```
/**
* Fonction de lecture de la température via un capteur DS18B20.
*/
byte getTemperature(float *temperature, byte reset_search) {
byte data[9], addr[8];
// data[] : Données lues depuis le scratchpad
// addr[] : Adresse du module 1-Wire détecté

/* Reset le bus 1-Wire ci nécessaire (requis pour la lecture du premier capteur) */
if (reset_search) {
ds.reset_search();
}

/* Recherche le prochain capteur 1-Wire disponible */
if (!ds.search(addr)) {
// Pas de capteur
return NO_SENSOR_FOUND;
}
crc8(addr, 7) != addr[7]) {
// Adresse invalide
return INVALID_ADDRESS;
}

/* Vérifie qu'il s'agit bien d'un DS18B20 */
if (addr[0] != 0x28) {
```

```
// Mauvais type de capteur

return INVALID_SENSOR;

}

/* Reset le bus 1-Wire et sélectionne le capteur */

ds.reset();

ds.select(addr);

/* Lance une prise de mesure de température et attend la fin de la mesure */

ds.write(0x44, 1);

delay(800);

/* Reset le bus 1-Wire, sélectionne le capteur et envoie une demande de lecture du scratchpad */

ds.reset();

ds.select(addr);

ds.write(0xBE);

/* Lecture du scratchpad */

for (byte i = 0; i < 9; i++) {

data[i] = ds.read();

}

/* Calcul de la température en degré Celsius */

*temperature = (int16_t) ((data[1] << 8)

/* Vérifie que l'adresse a été correctement reçue */
```

Revision #2
Created 5 February 2024 08:02:13 by Galleret Florent
Updated 5 February 2024 08:37:54 by Galleret Florent