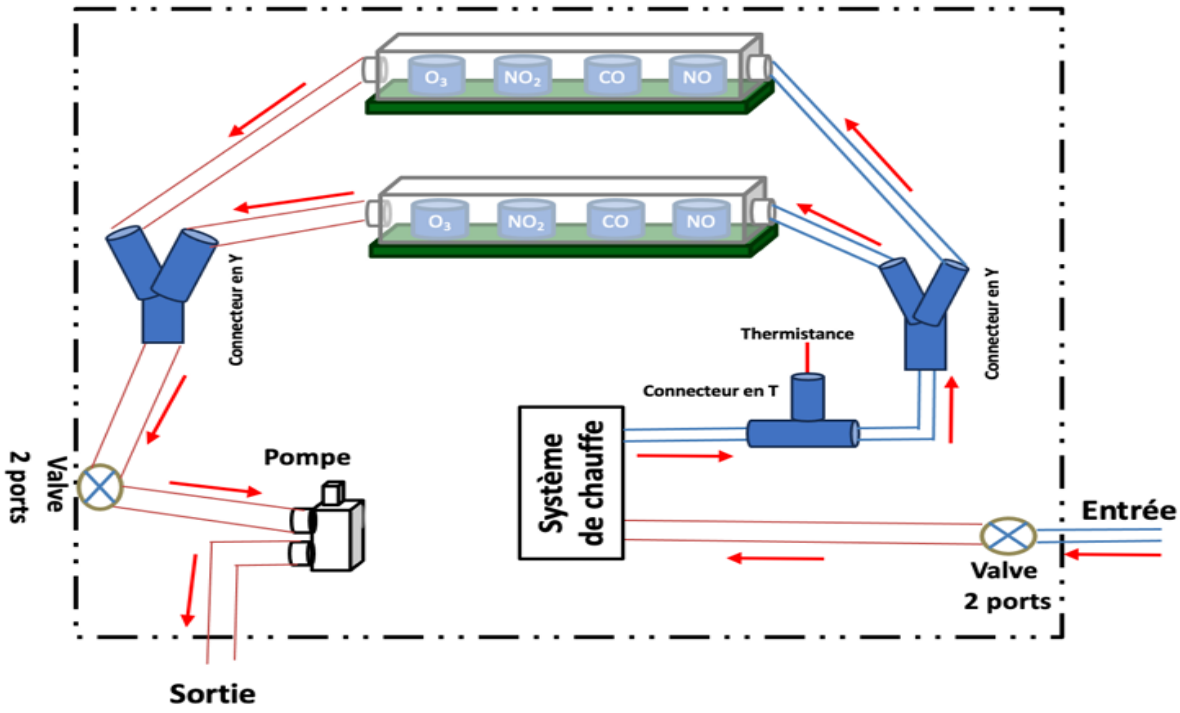
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES STRASBOURG

iCUBE

RAPPORT DE STAGE _2024-2025

Système Embarqué de Régulation Thermique pour l'Analyse de Gaz



Réalisé par :

BERMONE Aboubacar

Encadré par :

Mme. PIVERT Maya

M. LAROCHE Edouard

Licence 3 Mécatronique, Université de Strasbourg

Table des matières

I. Introduction

L’analyse de la qualité de l’air et la détection de gaz polluants nécessitent des conditions de mesure précises et reproductibles. Or, de nombreux capteurs de gaz sont sensibles aux variations de température et d’humidité, ce qui peut fausser leurs résultats. Un conditionnement thermique du flux d’air analysé est donc indispensable pour garantir la fiabilité des mesures.

Dans ce projet, nous avons conçu un système embarqué de régulation thermique destiné à stabiliser la température d’un fluide avant analyse. Le dispositif repose sur une résistance chauffante commandée par PWM via un MOSFET, la température étant mesurée à l’aide d’une thermistance NTC. Un Arduino UNO assure la régulation en temps réel, tandis que des capteurs complémentaires (SHT31, GPS BN220) et un écran LCD/OLED permettent de suivre et localiser les mesures.

La régulation mise en œuvre est proportionnelle simple (boucle P), afin d’obtenir un système stable, peu coûteux et facilement intégrable dans un dispositif embarqué. Ce travail s’inscrit dans une démarche complète allant de la simulation sous Proteus à la validation expérimentale, et constitue une étape clé vers la mise en place de systèmes complets de détection de gaz embarqués.

II. Contexte et Objectifs

2.1 Contexte

Les systèmes de mesure embarqués pour l'analyse de gaz trouvent aujourd'hui des applications variées :

- **Surveillance environnementale** : détection de polluants atmosphériques, suivi de la qualité de l'air en milieu urbain.
- **Sécurité civile** : aide aux pompiers et forces de secours pour évaluer la toxicité des fumées lors d'incendies.
- **Applications industrielles** : contrôle de procédés, prévention des fuites de gaz dangereux.
- **Recherche scientifique** : mesures en laboratoire ou sur drone pour cartographier des zones polluées.

Dans ces situations, la stabilité thermique des capteurs conditionne directement la qualité des données recueillies. Une variation de quelques degrés peut entraîner une dérive significative dans la réponse des capteurs électrochimiques ou semi-conducteurs. Il devient donc essentiel de maîtriser la température du flux d'air avant son analyse.

Ce projet se situe dans la continuité de travaux antérieurs menés à l'INSA Strasbourg, où l'intégration d'un système de chauffage et de régulation thermique a été identifiée comme un élément crucial pour améliorer la fiabilité des mesures.

2.2 Objectifs généraux

L'objectif principal du projet est de développer un système embarqué de régulation thermique assurant :

1. Le **chauffage contrôlé** d'un fluide à l'aide d'une résistance chauffante.
2. La **mesure et le suivi en temps réel** de la température grâce à une thermistance.
3. La **commande de puissance par PWM** via un MOSFET piloté par un microcontrôleur.
4. L'intégration de **capteurs complémentaires** (température/humidité, GPS) pour enrichir les données de mesure.
5. La **visualisation et validation** des résultats sur écran LCD/OLED et via simulation sous Proteus.

2.3 Cahier des charges technique

Le cahier des charges se traduit par les exigences suivantes :

Critère	Spécification	Justification
Plage de température visée	44–46 °C	Zone optimale pour limiter la dérive des capteurs de gaz
Précision de régulation	±1 °C	Suffisant pour assurer une stabilité des mesures
Microcontrôleur	Arduino UNO	Simplicité d'utilisation et large support logiciel
Actionneur	Résistance chauffante pilotée par MOSFET IRF7805	Fiabilité et robustesse
Capteurs intégrés	SHT31 (T°/HR), BN220 (GPS), thermistance NTC	Surveillance environnementale + géolocalisation
Mode de régulation	Boucle proportionnelle simple (P)	Compromis entre simplicité et efficacité
Affichage	Écran LCD ou OLED	Suivi en temps réel des mesures
Simulation	Proteus	Validation avant expérimentation
Alimentation	12 V (secteur ou batterie)	Compatibilité avec systèmes embarqués

Fonctionnement :

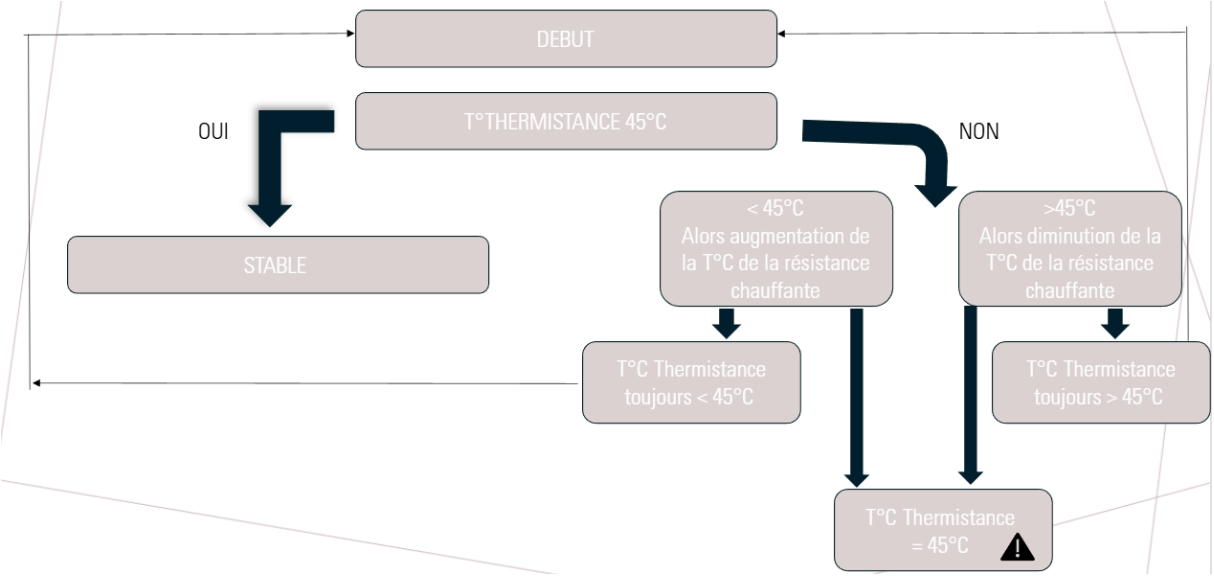


Figure 1: Graphe de Fonctionnement

2.4 Objectifs spécifiques

- **Court terme** :
  - Concevoir et simuler l'architecture électronique sous Proteus.
  - Implémenter un code Arduino permettant la régulation proportionnelle.
  - Vérifier expérimentalement la stabilité thermique obtenue.
- **Moyen terme** :
  - Intégrer le système dans une plateforme de mesure complète de gaz.
  - Étudier la consommation énergétique et l'autonomie sur batterie.
- **Long terme** :
  - Étendre la régulation à une boucle PID plus performante.
  - Réaliser une carte électronique dédiée (PCB) pour réduire l'encombrement.
  - Déployer le système sur un drone de détection de pollution.

III. Étude bibliographique et État de l'art

3.1 Importance du conditionnement thermique pour l'analyse de gaz

Les capteurs de gaz, qu'ils soient électrochimiques, semi-conducteurs ou optiques, présentent une forte dépendance aux conditions environnementales, en particulier à la température et à l'humidité.

- Les **capteurs électrochimiques** (Alphasense) nécessitent une température stable pour éviter la dérive du signal.
- Les **capteurs à semi-conducteur** (type MQ) voient leur sensibilité varier fortement avec la température ambiante.
- Les **capteurs optiques NDIR** (Non-Dispersive Infra-Red) présentent une dérive de l'absorption en fonction de la température de l'air.

Ainsi, un système de conditionnement thermique (chauffage + régulation) permet d'améliorer la reproductibilité et la fiabilité des mesures**.**

3.2 Techniques de régulation thermique existantes

Dans la littérature scientifique et technique, plusieurs approches sont utilisées pour réguler la température d'un fluide :

1. **Régulation On/Off**
   - Principe : la résistance est activée ou coupée en fonction d'un seuil.
   - Avantage : simplicité, faible coût.
   - Limite : oscillations importantes autour de la consigne.
2. **Régulation proportionnelle (P)**
   - Principe : la puissance de chauffe est proportionnelle à l'erreur entre consigne et valeur mesurée.
   - Avantage : stabilité correcte, mise en œuvre simple.
   - Limite : erreur statique résiduelle.
3. **Régulation PID (Proportionnel-Intégral-Dérivé)**
   - Principe : correction basée sur l'erreur instantanée, son intégrale et sa dérivée.
   - Avantage : très bonne précision et stabilité.
   - Limite : implémentation plus complexe, nécessite un calibrage fin.

Dans le cadre de ce projet, le choix s'est porté sur une régulation proportionnelle simple (boucle P), car elle représente un compromis efficace entre simplicité et performance.

3.3 Applications embarquées de régulation thermique

Des projets similaires ont déjà été développés dans le domaine des systèmes embarqués :

- **INSA Strasbourg – Projet PRT (2023/2024)** :
  Des étudiants ont conçu un capteur de pollution embarqué par drone, intégrant un système de conditionnement thermique pour stabiliser l'air avant analyse par des capteurs Alphasense.
- **Systèmes industriels** :
  Les analyseurs de gaz fixes (par exemple dans l'industrie chimique) intègrent presque toujours un four ou un dispositif de régulation thermique afin de stabiliser les gaz avant mesure.

- **Capteurs portatifs** :
  Certains détecteurs portables de gaz (ex. détecteurs de CO pour les pompiers) possèdent un micro-chauffage interne afin d'améliorer la sélectivité du capteur et limiter l'influence de l'humidité.

3.4 Comparaison des solutions

Méthode	Complexité	Précision	Coût	Utilisation typique
On/Off	Très faible	Faible	Très faible	Petits thermostats, chauffage domestique
Proportionnelle (P)	Faible	Moyenne	Faible	Régulation simple de température
PID	Moyenne à élevée	Très bonne	Moyen	Systèmes industriels, laboratoires
Régulation adaptative (fuzzy, prédictive)	Très élevée	Excellente	Élevé	Recherche, systèmes hautement critiques

3.5 Conclusion de l'étude bibliographique

L'analyse de l'état de l'art montre que :

- La régulation thermique est indispensable pour fiabiliser l'analyse des gaz.
- Les solutions existantes vont de la simple régulation On/Off jusqu'aux régulations PID complexes.
- Pour un système embarqué, compact et peu coûteux, la régulation proportionnelle par PWM + MOSFET constitue une solution pertinente.

IV. Théorie et Bases Physiques

4.1 Transferts thermiques

Le chauffage d'un fluide ou d'un élément solide met en jeu différents phénomènes physiques :

- **Conduction** : transfert de chaleur à travers un matériau solide.
  Loi de Fourier :

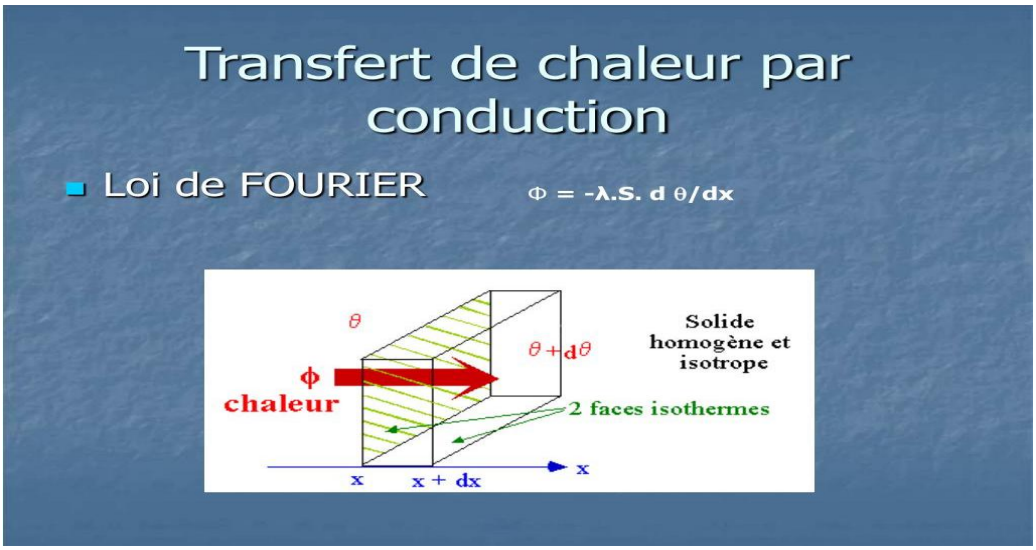


Figure 2:Conduction

- **Convection** : échange de chaleur entre une surface et un fluide en mouvement.
  Loi de Newton :

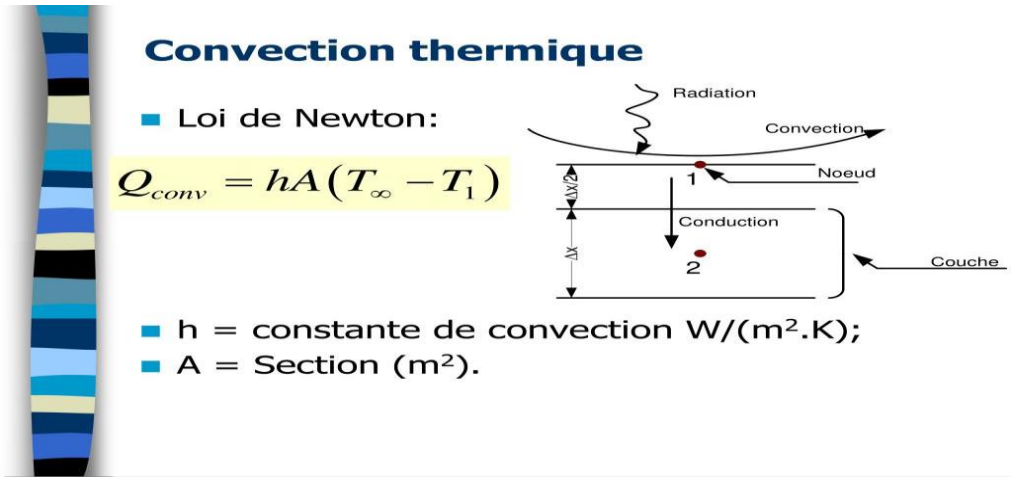


Figure 3: Convection

- **Rayonnement** : émission d'énergie sous forme de rayonnement électromagnétique.
  Loi de Stefan-Boltzmann :

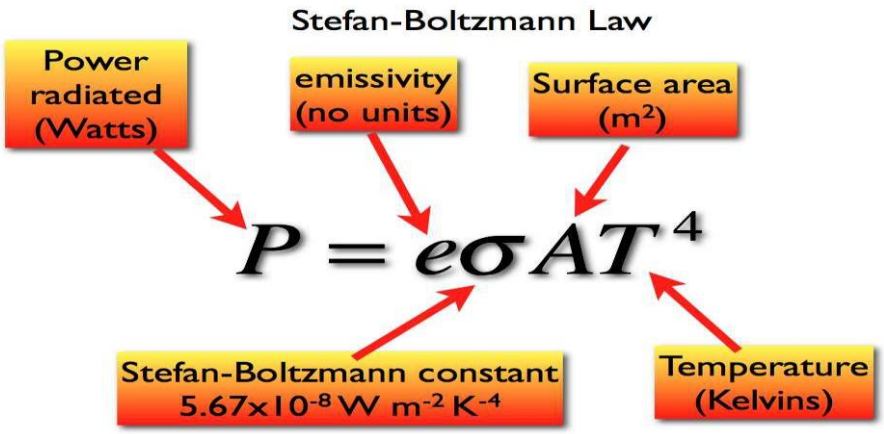


Figure 4: Rayonnement

Dans le cas du système étudié, le transfert dominant est la conduction dans la résistance chauffante, puis la convection de la chaleur vers l'air à chauffer.

4.2 Thermistance

La thermistance est un capteur de température résistif dont la valeur de résistance varie en fonction de la température.

- **NTC (Negative Temperature Coefficient) :** la résistance diminue quand la température augmente.
- **PTC (Positive Temperature Coefficient)** : la résistance augmente avec la température.

La relation résistance-température pour une thermistance NTC est donnée par l'équation de Steinhart-Hart :

$$\frac{1}{T} = A + B.\ln(R) + C.(\ln(R))^3$$

Où :

- T est la température en Kelvin,
- R la résistance de la thermistance,
- A, B, C sont des constantes dépendant du matériau.

En pratique, la thermistance est utilisée dans un pont diviseur de tension :

$$V_{out} = V_{cc}.\frac{R_{NTC}}{R_{fixe} + R_{NTC}}$$

Cette tension est ensuite lue par l'entrée **ADC** de l'Arduino, permettant de calculer la température.

4.3 MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor)

Le MOSFET est un transistor de puissance largement utilisé comme interrupteur électronique.

- Dans ce projet, un MOSFET canal N (IRF7805) est utilisé pour contrôler le courant traversant la résistance chauffante.
- Commande : le MOSFET est piloté par une tension appliquée à sa grille (**Vgs**).
- Fonctionnement en mode interrupteur :
  - Si Vgs<Vth → MOSFET bloqué, pas de courant.
  - Si Vgs>Vth → MOSFET saturé, conduction quasi parfaite.

La résistance interne à l'état passant, appelée **Rds(on)**, doit être faible pour limiter les pertes par effet Joule :

$$\boldsymbol{P_{pertes} = I^2 . Rds_{(on)}}$$

Dans notre système, le MOSFET est commandé en PWM pour moduler la puissance transmise à la résistance chauffante.

4.4 PWM (Pulse Width Modulation)

La PWM est une technique de commande qui consiste à appliquer une tension continue en impulsions rapides, dont la largeur est modulée en fonction du signal de commande.

- **Fréquence PWM** : suffisamment élevée pour que le chauffage soit perçu comme une puissance moyenne (généralement quelques kHz).
- **Rapport cyclique (Duty Cycle)** :

$$\boldsymbol{D = \frac{T_{on}}{T_{période}} \times 100}$$

  - D= =0% : chauffage coupé.
  - D=100% : chauffage maximal.
  - D=50%: puissance moyenne à 50 %.

La puissance moyenne transmise à la charge est :

$$\mathbf{Pmoy = D \cdot Pmax}$$

C’est cette modulation qui permet d’ajuster finement la température du fluide.

4.5 Régulation proportionnelle (boucle P)

Un système de régulation vise à maintenir une grandeur (ici la température) proche d’une consigne fixée.

La commande proportionnelle est définie par :

$$u(t)=Kp\cdot e(t)$$

Avec :

- u(t): signal de commande (ici le rapport cyclique PWM),
- Kp : gain proportionnel,
- e(t) : erreur = (Tconsigne – Tmesurée).

Avantages :

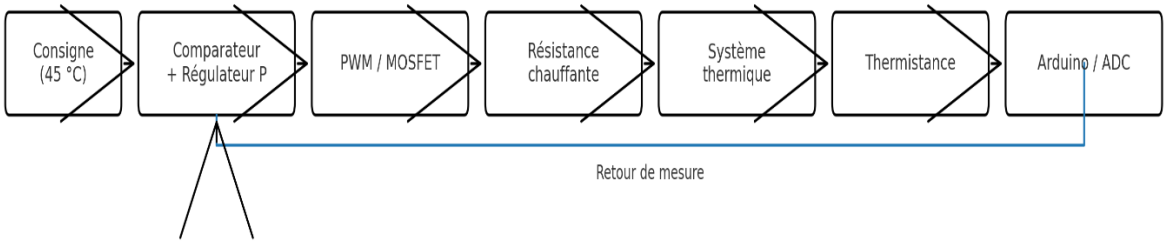
- Simplicité de mise en œuvre.
- Stabilité correcte si Kp est bien choisi.

Limites :

- Existence d’une erreur statique (la température peut osciller légèrement autour de la consigne).
- Risque d’oscillations si Kp est trop élevé.

4.6 Schéma fonctionnel global

Le système peut être représenté par un **bloc d’asservissement** :



Ce schéma illustre la **boucle fermée** qui assure la régulation thermique.

V. Présentation des Composants

5.1 Arduino UNO

L'Arduino UNO est un microcontrôleur basé sur l'ATmega328P. Il constitue le cœur du système de régulation thermique.

- **Rôle** : acquisition des données capteurs (thermistance, SHT31, GPS), génération du signal PWM pour piloter le MOSFET, gestion de l'affichage et enregistrement des mesures.
- **Caractéristiques principales** :
  - 14 entrées/sorties numériques, dont 6 capables de générer du PWM.
  - 6 entrées analogiques (10 bits ADC, soit 1024 niveaux de résolution).
  - Fréquence d'horloge : 16 MHz.
  - Tension d'alimentation : 5 V.
  - Communication : I²C, UART, SPI.

Figure 5: Arduino UNO

- **Justification du choix** : plateforme simple, bien documentée, adaptée aux prototypes rapides.

5.2 Thermistance NTC

La thermistance NTC (Negative Temperature Coefficient) est utilisée comme capteur de température dans le système.

- **Principe** : sa résistance diminue lorsque la température augmente.
- **Caractéristiques typiques** :
  - Valeur nominale : 10 kΩ à 25 °C.
  - Plage de mesure : –40 °C à +125 °C.
  - Constante Beta (B) : ~3950 K.

- **Rôle dans le système** : mesure en temps réel de la température du fluide chauffé.
- **Mise en œuvre** : intégrée dans un pont diviseur de tension, la tension de sortie est lue par l'ADC de l'Arduino.
- **Justification** : faible coût, précision suffisante, temps de réponse rapide.

Figure 6: Thermistance

5.3 Résistance chauffante

La résistance chauffante est l'élément actif qui élève la température du fluide.

- **Principe** : dissipation d'énergie électrique sous forme de chaleur (effet Joule).

  P=U2RouP=R·I2P = \frac{U^2}{R} \quad \text{ou} \quad P = R \cdot I^2P=RU2 ouP=R·I2

- **Caractéristiques typiques** :
  - Tension nominale : 12 V.
  - Puissance : 5–20 W (selon modèle).
  - Matériau : fil résistif (nichrome).
- **Justification** : technologie simple, robuste, facilement pilotable via un MOSFET.

Figure 7: Resistance chauffante

5.4 MOSFET IRF7805

Le MOSFET IRF7805 est utilisé pour commuter la puissance fournie à la résistance chauffante.

- **Caractéristiques principales** :
  - Type : MOSFET canal N.
  - Tension drain-source max : 55 V.
  - Courant drain continu : 80 A.
  - Rds(on) : ~0,009 Ω.
- **Rôle** : interrupteur électronique commandé par l'Arduino (PWM).
- **Justification** : supporte largement le courant nécessaire, faible dissipation, fiable pour un système embarqué.

5.5 Capteur SHT31

Le SHT31 est un capteur numérique d'humidité et de température.

- **Caractéristiques principales** :
  - Interface : I²C.
  - Précision en température : ±0,3 °C.
  - Précision en humidité relative : ±2 %.
  - Plage : –40 °C à +125 °C.
- **Rôle** : mesurer les conditions environnementales et valider la régulation.
- **Justification** : permet de corriger les mesures du système en tenant compte des conditions ambiantes.

Figure 8: SHT31

5.6 GPS BN220

Le GPS BN220 est un module de géolocalisation compact basé sur le système Beidou/GPS.

- **Caractéristiques principales** :
  - Communication : UART (9600 bauds).

- Précision : <2 m (après fix).
- Temps d'acquisition : 30 s à froid, <1 s à chaud.

- **Rôle** : associer les mesures de gaz à une localisation géographique.
- **Justification** : faible coût, compact, compatible Arduino.

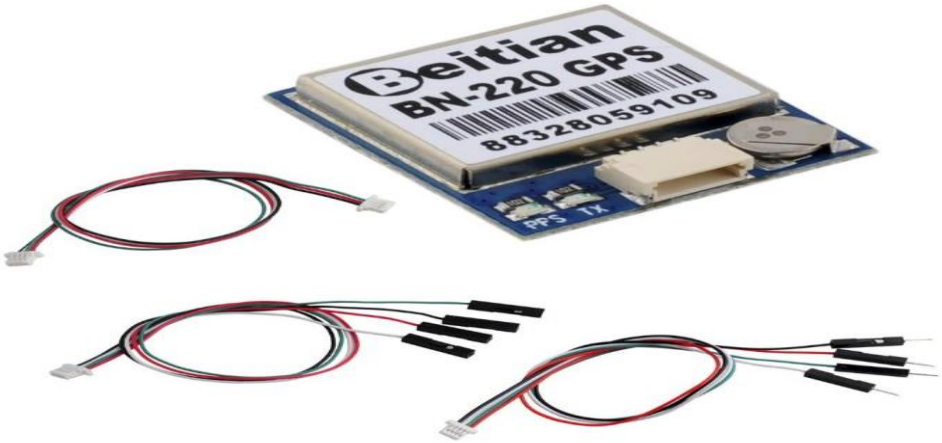


Figure 9: BN220

5.7 Écran LCD / OLED

Un écran LCD 16x2 ou un OLED I²C est utilisé pour l'affichage des mesures en temps réel.

- **Rôle** : afficher la température mesurée, le rapport cyclique PWM, l'humidité, et la position GPS.
- **Caractéristiques typiques** :
  - LCD : 16 colonnes × 2 lignes, alimentation 5 V.
  - OLED : résolution 128x64, interface I²C, faible consommation.
- **Justification** : interface utilisateur simple et claire.

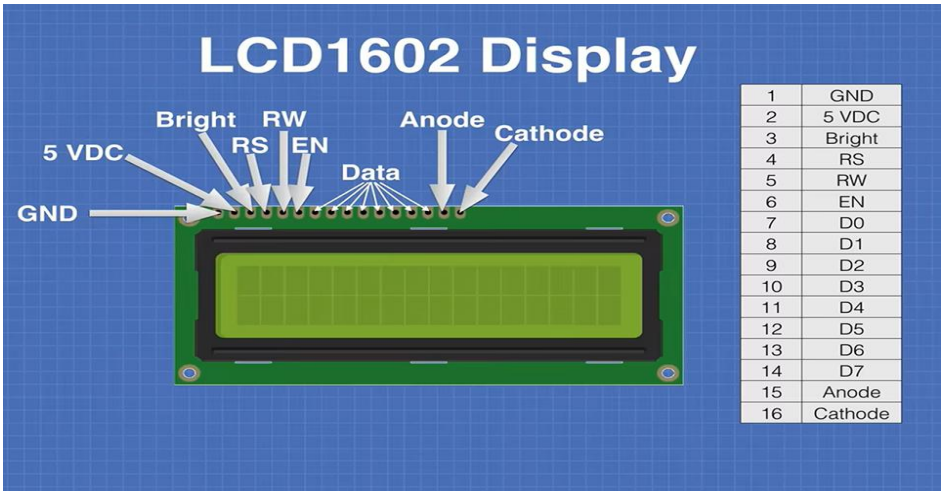


Figure 10: Ecran LCD

5.8 Alimentation

Le système est alimenté en 12 V, avec conversion interne pour fournir le **5 V** à l'Arduino et aux capteurs.

- **Rôle** : fournir l'énergie nécessaire à la résistance chauffante et à l'électronique.
- **Justification** : standard des systèmes embarqués (compatible batteries Li-Ion 3S).

Figure 11: Alimentation

VI. Architecture Électronique et Logicielle

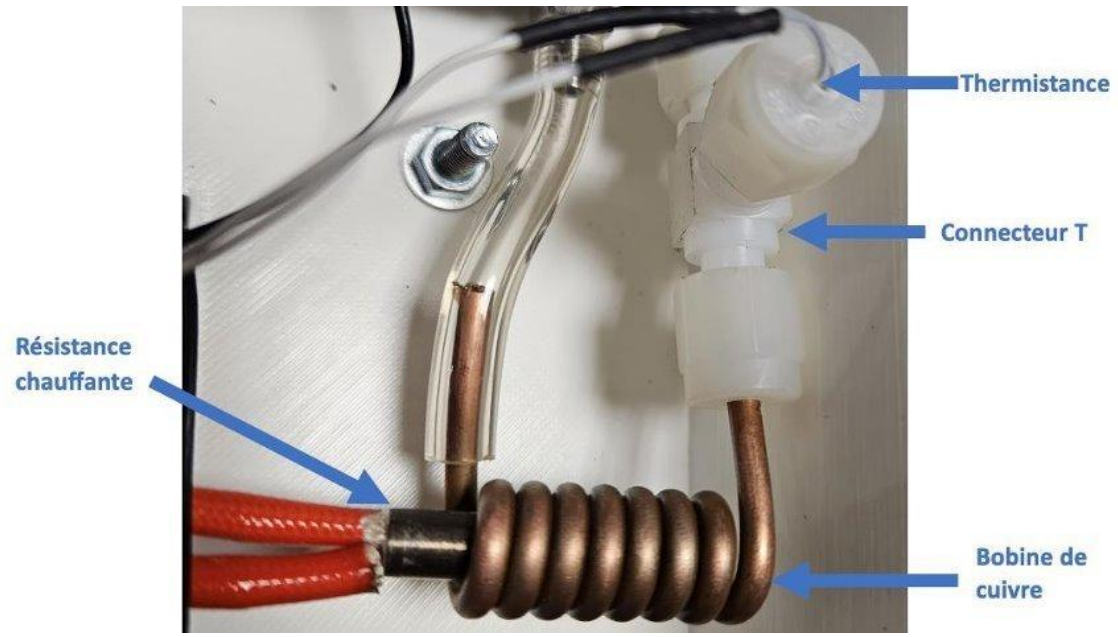


Figure 12: Schéma du système

6.1 Schéma fonctionnel global

Le système de régulation thermique se compose de plusieurs sous-systèmes interconnectés :

- **Sous-système de mesure** : thermistance NTC, capteur SHT31.
- **Sous-système de commande** : Arduino UNO.
- **Sous-système de puissance** : MOSFET IRF7805 + résistance chauffante.
- **Sous-système d'affichage** : écran LCD/OLED, LED indicatrice.
- **Sous-système de localisation** : GPS BN220.
- **Alimentation** : source 12 V, régulée pour alimenter l'Arduino en 5 V.

Description :
La thermistance envoie une tension proportionnelle à la température via un pont diviseur. L'Arduino lit cette tension avec son ADC, calcule la température et compare à la consigne. En fonction de l'écart, il ajuste le rapport cyclique PWM appliqué à la grille du MOSFET, qui module la puissance dissipée dans la résistance chauffante. Le SHT31 fournit en parallèle la température/humidité ambiante, et le GPS ajoute les coordonnées. L'ensemble est affiché sur un écran LCD/OLED.

6.2 Schéma électronique

Le schéma complet (réalisé sur **Proteus**) se compose de :

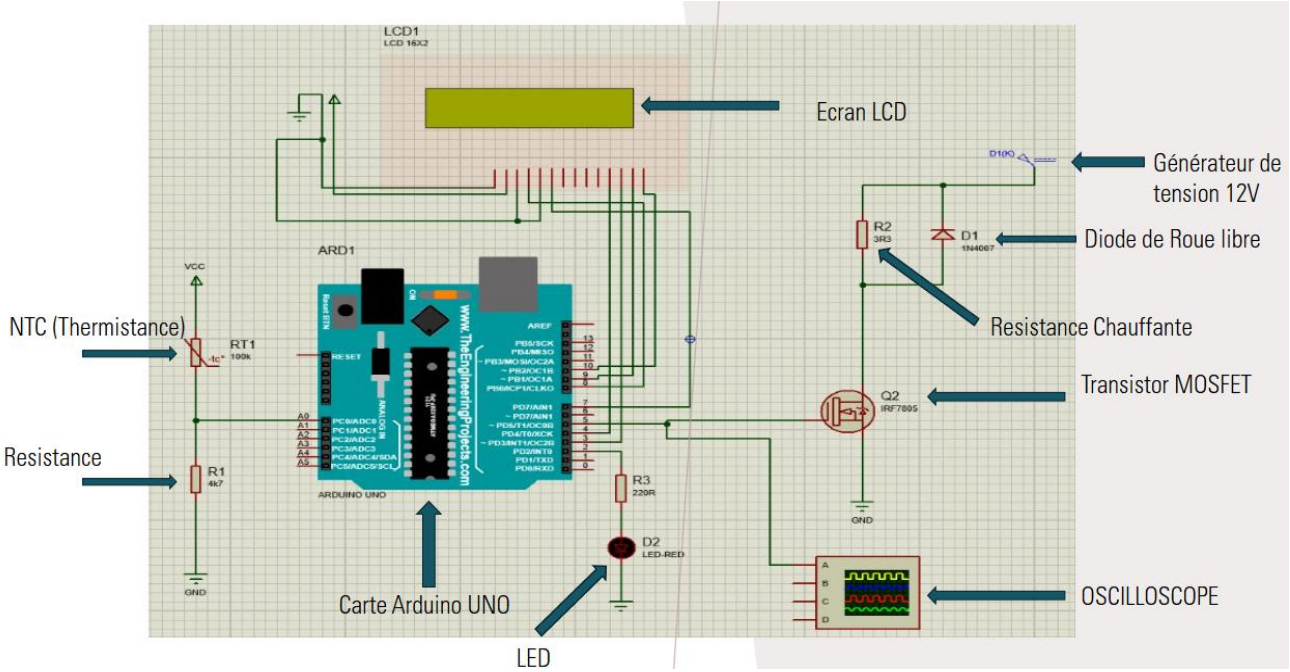


Figure 13: Schéma sur Proteus

6.3 Logique logicielle

Le programme Arduino suit une structure simple mais efficace :

Étapes principales :

1. **Initialisation** : configuration des broches, initialisation de l'I²C, UART, LCD/OLED.
2. **Lecture capteurs** : thermistance (ADC), SHT31 (I²C), GPS (UART).
3. **Calcul température** : conversion ADC → résistance NTC → température (via équation Beta ou Steinhart-Hart).
4. **Comparaison avec consigne** : calcul de l'erreur $e(t)=T_{consigne}-T_{mesure}$
5. **Application régulation P** :

   $PWM=K_p \cdot e(t)$

   (Limité entre 0 % et 100 %).

6. **Commande MOSFET** : génération du signal PWM sur la broche dédiée.
7. **Affichage** : envoi des données vers l'écran LCD/OLED.
8. **Boucle infinie** : répétition en temps réel.

6.4 Bibliothèques utilisées

- **Wire.h** : communication I²C.
- **Adafruit_SHT31.h** : lecture du capteur SHT31.
- **TinyGPS++** : décodage des trames GPS.
- **LiquidCrystal.h** ou **Adafruit_SSD1306.h** : gestion de l'écran LCD/OLED.

VII. Dimensionnement du Système

7.1 Besoin en chauffage

La puissance nécessaire pour élever la température d'un fluide dépend de sa masse, de sa capacité thermique massique et de la différence de température souhaitée.

Formule générale :

$$Q = m.c.\Delta T$$

$$P = \frac{Q}{t}$$

Avec :

- m= masse du fluide chauffé (kg),
- c= capacité thermique massique (J/kg·K),
- ΔT = élévation de température (K),
- t= temps de chauffage (s).

Exemple pour l'air :

- Masse d'air chauffée dans un petit volume ≈ 0,002 kg.
- Cair≈1000 J/kg\Kc
- ΔT=20°C (passage de 25 °C à 45 °C).
- Temps de chauffe visé = 30 s.

Q=0,002×1000×20=40

P=40/30≈1,3 W

Donc, une résistance chauffante de 5 W est suffisante pour ce type d'application, car elle couvre les pertes thermiques par convection et rayonnement.

7.2 Choix de la résistance chauffante

La puissance dissipée par une résistance est donnée par :

$$P = \frac{U^2}{R}$$

Si on choisit une résistance de R=27 Ω alimentée en 12 V :

$P=12^2/27 \approx 5,3$ W

Ce choix correspond bien au besoin calculer précédemment.

7.3 Dimensionnement du MOSFET

Le MOSFET IRF7805 doit supporter :

- La tension drain-source VDS=12 V
- Le courant drain :

  I=U/R=12/27≈0,44 A

Le courant maximal est **0,5 A**, très inférieur à la capacité du MOSFET (80 A).

Pertes par conduction

$$P_{cond} = I^2 . R_{ds(on)}$$

Avec Rds(on)≈ 0,009 Ω :

$$\mathbf{Pcond=0,442\times0,009\approx0,0017\ W}$$

Les pertes sont négligeables, le MOSFET ne chauffe presque pas.

7.4 Alimentation électrique

- **Puissance totale consommée** :
  - Chauffage : 5 W max.
  - Arduino + capteurs + écran : ~0,5 W.
  - Total ≈ **5,5 W**.
- **Batterie embarquée** : par exemple une Li-Ion 3S (11,1 V, 2200 mAh).
  - Énergie disponible :

$$\mathbf{E=U\cdot C=11,1\cdot2,2=24,4\ Wh}$$

Autonomie estimée :

$$t = \frac{E}{P} = \frac{24,4}{5,5} \approx 4,4h$$

Le système peut fonctionner environ 4 heures en autonomie avec une petite batterie de drone.

III. Simulation et Expérimentation

8.1 Simulation sous Proteus

8.1.1 Objectifs de la simulation

La simulation a permis de valider le comportement du système avant montage réel, notamment :

- Le câblage correct des composants,
- La réponse de la boucle de régulation proportionnelle,
- La variation du signal PWM en fonction de la température mesurée,
- L'affichage des informations sur l'écran LCD.

8.1.2 Schéma de simulation

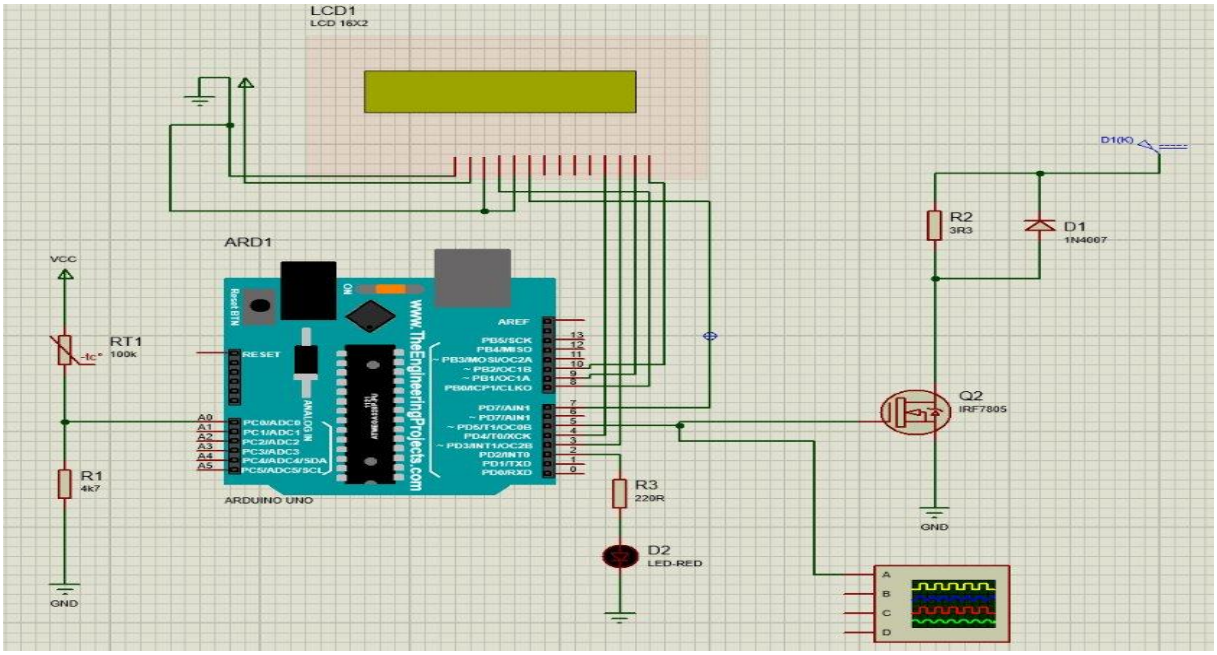


Figure 14: schéma de simulation

Description visuelle :

- La thermistance est intégrée dans un pont diviseur et reliée à A0.
- Le MOSFET commute la charge en fonction du signal PWM généré sur D5.
- L'écran affiche en temps réel la température et le rapport cyclique.

8.2 Résultats de la simulation

8.2.1 Évolution de la température

- La température simulée monte progressivement jusqu'à la zone de consigne (44–46 °C).
- Le système stabilise la température sans oscillations trop importantes.

8.2.2 Signal PWM

- Pour T<44°, le PWM est élevé (~200–255/255).
- Pour 44–46°C, le PWM se stabilise autour d'une valeur intermédiaire.
- Pour T>46°C, le PWM tombe à zéro → chauffage coupé.

Cela confirme que la logique de régulation proportionnelle est bien implémentée.

8.2.3 Affichage LCD

- Température mesurée (°C).
- Rapport cyclique PWM en %.
- L'évolution en temps réel est lisible et claire.

8.3 Expérimentation réelle

8.3.1 Montage pratique

Un prototype a été assemblé avec :

- Arduino UNO,
- thermistance NTC,
- résistance chauffante 12 V / 5 W remplacé par une led,
- MOSFET IRFZ44R,
- écran OLED I²C.
- capteurs SHT31 et GPS BN220,

L'ensemble a été alimenté par une alimentation de laboratoire réglée à 12 V.

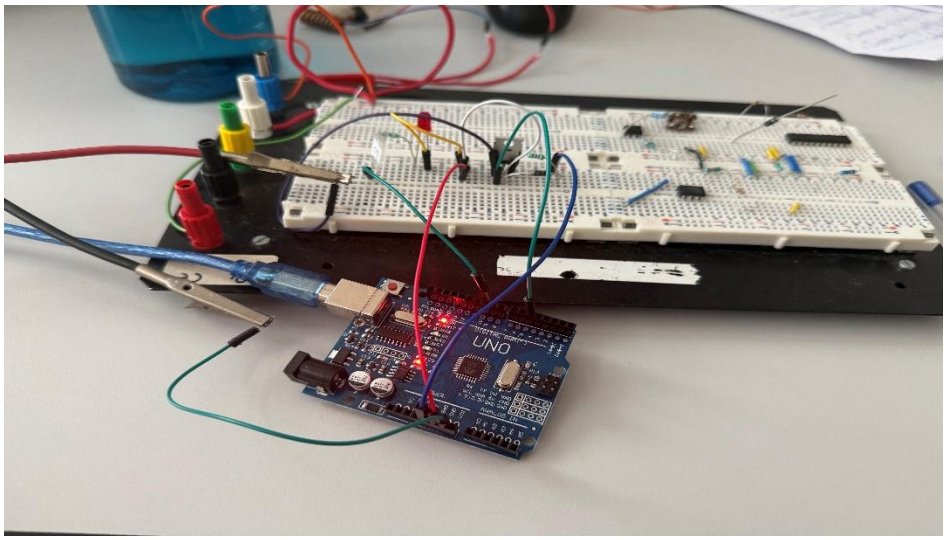


Figure 15: Test sur plaquette avec une led

8.3.2 Résultats expérimentaux

- **Signal PWM observé à l'oscilloscope** : fréquence stable (~490 Hz), rapport cyclique variable selon la température.

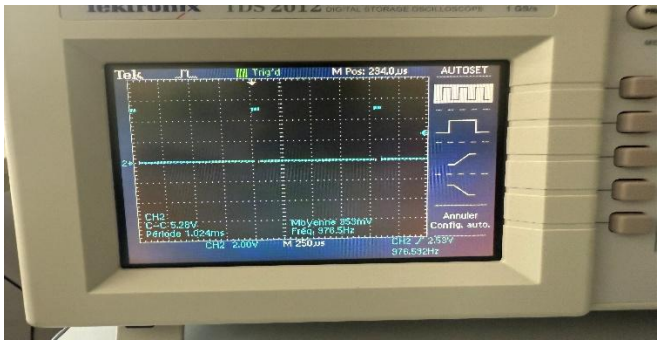


Figure 16: signal PWM

- **Capteur SHT31** : mesure concordante avec la thermistance

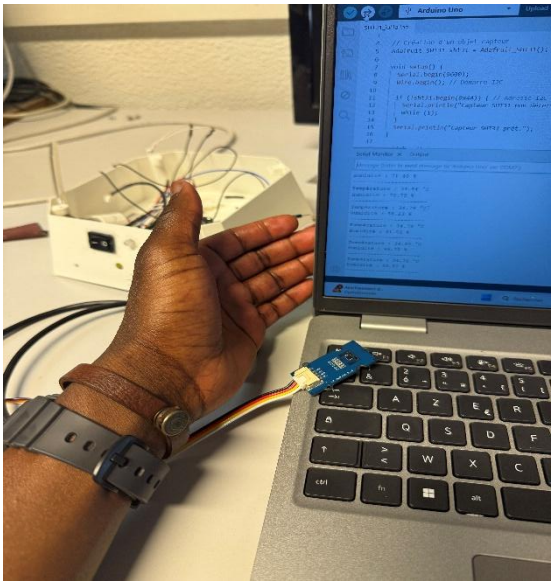


Figure 18: TEST DU SHT31 sec

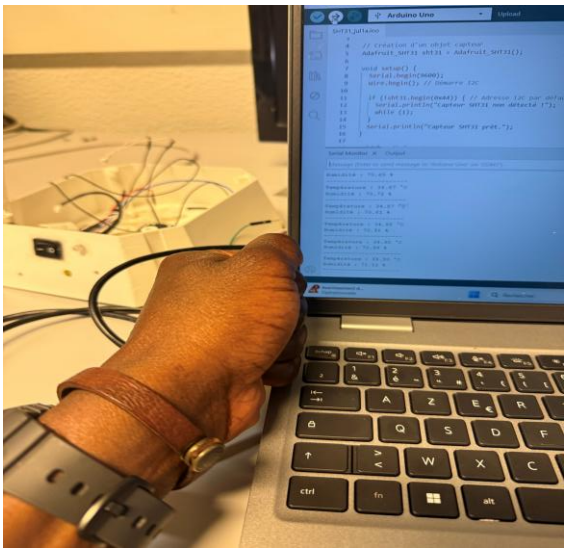


Figure 17: TEST du SHT31 Humide

- **GPS BN220** : acquisition de position correcte après 1 minute à ciel ouvert.

8.3.3 Limitations rencontrées

- **Surchauffe du MOSFET** (légère) → dissipateur ajouté.
- **Lecture bruitée de la thermistance** → filtrage logiciel (moyenne glissante).
- **Fix GPS lent en intérieur** → nécessité de travailler à ciel ouvert.

IX. Discussion, Limites et Améliorations

9.1 Écarts simulation ↔ réel : analyse critique

- **Inertie thermique sous-estimée** : la montée en T est plus lente en vrai (pertes par convection/rayonnement).
  → intégrer un **coefficient global de pertes** (h·S) dans le modèle et recalibrer Kp.
- **Bruit de mesure NTC** (ADC 10 bits, pont diviseur) : micro-oscillations de PWM autour de la consigne.
  → **filtrage numérique** (moyenne glissante N=8–16) et/ou **filtre RC** (R=1–4.7 kΩ, C=100 nF) avant l'ADC.
- **Surchauffe locale du MOSFET** lors de longues phases à haut duty.
  → dissipateur + pâte thermique ; vérifier Pcond=I^2Rds(on) et espacer la carte de la source chaude.

9.2 Régulation : du P simple vers mieux

9.2.1 P avec bande morte & hystérésis (simple, efficace)

- **Bande morte** ±0,3–0,5 °C autour de 45 °C → évite le "pompage".
- **Hystérésis** sur la décision ON/OFF au voisinage des seuils 44–46 °C.

9.2.2 Ajout d'un terme intégral léger (PI)

$$U = K_p e + K_i \int e dt$$

- **But** : supprimer l'erreur statique.
- **Conseil** : Ki petit (ex. 0.01–0.05 s^{-1}) + anti-windup (clamp de l'intégrale lorsque PWM=0 % ou 100 %).
- **Pseudocode anti-windup** : n'intégrer que si 0<PWM<2550.

9.2.3 Dérivée filtrée (PID "PD-lite")

$$U = K_p e + K_d \frac{d_e}{dt}$$

- **But** : anticiper les sur-élèvements quand l'inertie est faible.
- **Filtrage** : dérivée sur erreur filtrée (différence d'une moyenne glissante) pour éviter d'amplifier le bruit.

9.2.4 Fréquence PWM et pertes

- **Arduino Uno** : ~490 Hz par défaut.
- Pour réduire le ripple thermique audible et les pertes de commutation, viser 1–4 kHz (timer config), ça reste OK pour une charge résistive.

9.3 Mesure & calibration

9.3.1 Calibration NTC

- Méthode **β** (rapide) ou Steinhart-Hart (plus précise).
- Procédure à 3 points (ex. 25 °C, ~45 °C, ~60 °C) avec thermomètre de référence → ajuste A, B, C (ou β) et stocke en EEPROM.

9.3.2 Linéarisation & résolution

- Choix de Rfixe ≈ R_NTC (45 °C) pour maximiser la sensibilité ADC autour de la consigne.
- Option : moyenne pondérée (ex. 70 % NTC, 30 % SHT31) si les sondes sont co-localisées et cohérentes.

9.4 Optimisation énergétique (embarqué/batterie)

- **Modulation de puissance** : clamp duty max juste suffisant (ex. 70–80 %) après la phase de démarrage.
- Sleep modes Arduino entre acquisitions (≥100 ms) ; OLED off si inactif.
- Préférer un buck (rendement 90–95 %) à un LDO pour descendre 12 V→5 V si >150 mA.

9.5 Données & traçabilité

- Logging sur carte SD (T_NTC, T/H SHT31, duty, GPS, timestamp) à 1–2 Hz.
- Format CSV → analyse facile (courbes T(t), duty(t), dérive jour/nuit).
- Ajout d'un checksum simple (CRC-8) pour fiabiliser les logs.

9.8 Méthode de tuning recommandée (pratique)

1. Fixer consigne 45 °C, Ki=Kd=0.
2. Augmenter Kp jusqu'à un léger dépassement (<1 °C).
3. Introduire bande morte ±0,3 °C.
4. Ajouter Ki très faible pour supprimer l'écart statique (vérifier que l'oscillation ne réapparaît pas).
5. Optionnel : Kd minimal si sur-élévation >1,5 °C.

9.9 Tableau récap des problèmes & solutions

Problème	Cause probable	Solution rapide	Solution durable
Oscillation autour de 45 °C	Kp trop élevé, bruit NTC	Bande morte + filtrage	PI avec anti-windup
Lenteur à l'allumage	Inertie + pertes	Duty boost 100 % 5–10 s	Isolation + répartiteur
Mesure instable	Bruit ADC, câblage	Moyenne N=8–16	Filtre RC + routage propre
MOSFET tiède	Ventilation nulle	Petit dissipateur	Choix Rds(on) plus bas, boîtier D-PAK
Autonomie faible	Perte régulation 12→5 V	Duty clamp, sleep MCU	Convertisseur buck haut rendement
GPS lent en intérieur	Ciel masqué	Mise en place extérieure	Antenne active, warm-start

9.10 Conclusion de la discussion

La boucle P actuelle répond au cahier des charges (±1 °C). Pour gagner en précision et robustesse en conditions réelles variées, la trajectoire logique est **:** P + bande morte → PI anti-windup → PD léger si besoin, en parallèle d'une calibration NTC propre, d'un petit refactoring hardware (découplage, mécanique) et d'une gestion énergie optimisée.

X. Perspectives et Feuille de route

10.1 Amélioration de la régulation

- Passage d'une régulation P simple vers un PID complet pour atteindre une précision de ±0,2 °C.
- Intégration d'un auto-tuning logiciel (méthode de Ziegler-Nichols ou relay feedback) pour ajuster automatiquement les coefficients Kp,Ki,Kd.
- Étude d'une commande prédictive (Model Predictive Control) pour mieux anticiper l'inertie thermique.

10.2 Optimisation matérielle

- Conception d'un PCB dédié intégrant MOSFET, capteurs, connecteurs et régulation de puissance → plus compact et robuste qu'une breadboard.
- Ajout d'un capteur de courant (INA219) pour monitorer la consommation et éviter la surcharge.
- Choix de MOSFET à Rds(on) plus faible (IRLZ44N par exemple) pour améliorer l'efficacité.

10.3 Communication et intégration

- Transmission sans fil (LoRa, WiFi, BLE) pour envoyer en temps réel les mesures à une station au sol.
- Intégration dans une plateforme drone avec synchronisation des données GPS et capteurs environnementaux.
- Mise en place d'une base de données en ligne (cloud) pour stocker et visualiser les mesures.

10.4 Optimisation énergétique

- Passage de l'Arduino UNO à un ESP32 : consommation réduite, WiFi/Bluetooth intégrés, puissance de calcul supérieure.
- Utilisation de modes deep sleep entre acquisitions pour prolonger l'autonomie.
- Alimentation via un buck converter haut rendement pour passer de 12 V à 5 V.

10.5 Déploiement futur

- Miniaturisation et encapsulation du système dans un boîtier résistant aux conditions extérieures.
- Développement d'une version multi-capteurs (CO, NO_2, O_3, PM2.5) pour une analyse de pollution plus complète.
- Validation dans des campagnes de mesure terrain (zones urbaines, industrielles, rurales).

XI. Conclusion

Ce projet a permis de concevoir, simuler et tester un système embarqué de régulation thermique destiné à l'analyse de gaz.
Les principaux résultats obtenus sont :

- La mise en place d'une boucle de régulation proportionnelle simple mais efficace, stabilisant la température entre 44 °C et 46 °C.
- La validation sous Proteus puis en expérimentation pratique avec des résultats concordants.
- Une architecture modulaire combinant mesure (NTC, SHT31), commande (Arduino + MOSFET) et affichage (LCD/OLED).
- Un dimensionnement correct (résistance 5 W, MOSFET largement surdimensionné, autonomie ~4 h).

Le système répond aux objectifs fixés et constitue une base solide pour des améliorations futures (PID, optimisation énergétique, intégration drone).

XII. Annexes

A. Code Arduino

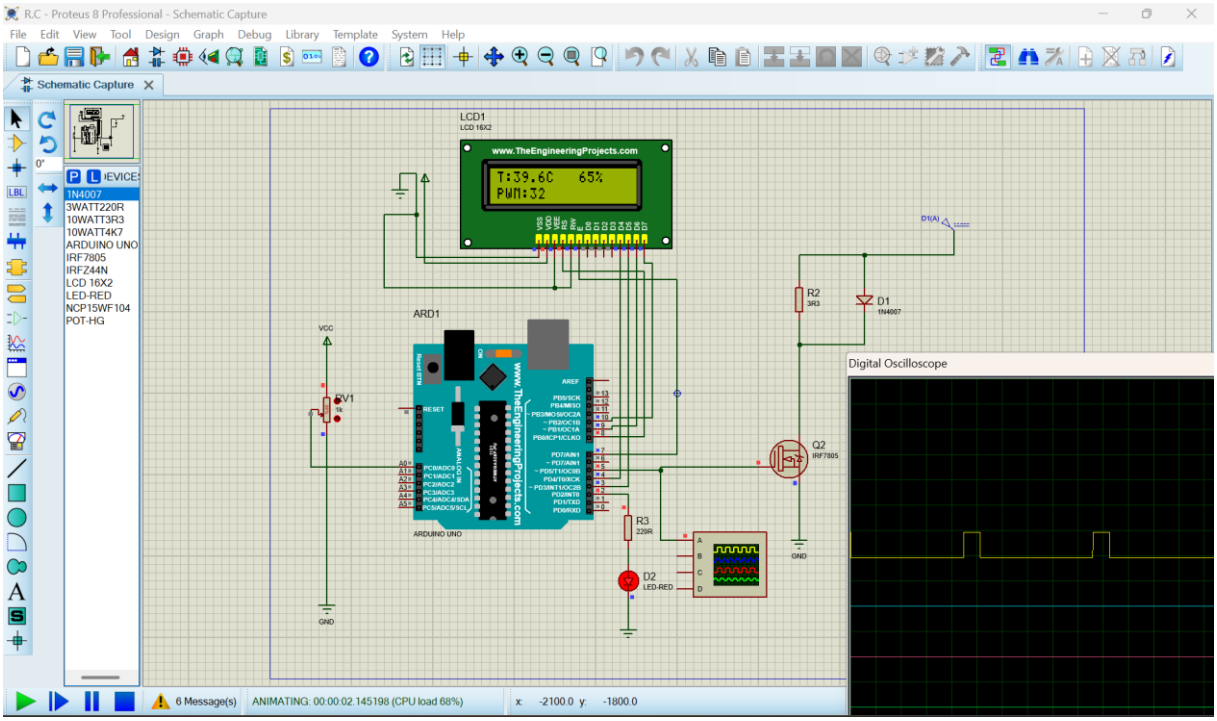
```
#include <LiquidCrystal.h>

// LCD : RS, E, D4, D5, D6, D7
LiquidCrystal lcd(8, 7, 4, 3, 9, 10);

// Brochage logique
int potPin = A0;
int pwmPin = 5;
int ledPin = 2;

// Calibration thermique
float tempMax = 60.0; // À curseur bas (% = 0)
float tempMin = 0.0;  // À curseur haut (% = 100)

// Hystérésis
float seuilBas = 44.0;   // Allume la LED si T < 44°C
float seuilHaut = 46.0;  // Éteint la LED si T > 46°C
int PWM_MAX = 255;
int PWM_MIN = 0;
bool ledEtat = false
float T_previous = 0;
unsigned long stableStartTime = 0;
bool tempStable = false;
bool ledAutoOff = false;

void setup() {
```

```
  pinMode(potPin, INPUT);
  pinMode(pwmPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  analogWrite(pwmPin, PWM_MIN);
  lcd.begin(16, 2);
  lcd.print("Systeme actif...");
  delay(1000);
  lcd.clear();
}

void loop() {
  // Lecture potentiomètre
  int potValue = analogRead(potPin);
  float percent = 100.0 - (potValue * 100.0 / 1023.0);
  float temperature = tempMin + ((percent / 100.0) * (tempMax - tempMin));

  // Régulation PWM
  float erreur = seuilHaut - temperature;
  int pwm = erreur * 5.0;
  pwm = constrain(pwm, PWM_MIN, PWM_MAX);
  analogWrite(pwmPin, pwm);

  // Détection stabilité température
  if (abs(temperature - T_previous) < 0.2) {
    if (!tempStable) {
      tempStable = true;
      stableStartTime = millis();
  }
  } else {
```

```
  tempStable = false;
  stableStartTime = 0;
  ledAutoOff = false;  // Réactive la LED si variation
}

// Gestion LED avec hystérésis (bloquée si extinction auto déjà faite)
if (!ledAutoOff) {
  if (temperature < seuilBas) {
    ledEtat = true;
  } else if (temperature > seuilHaut) {
    ledEtat = false;
  }
}

// Extinction auto après 5s de stabilité
if (ledEtat && tempStable && !ledAutoOff) {
  if (millis() - stableStartTime >= 5000) {
    ledEtat = false;
    ledAutoOff = true;  // Éviter de réallumer la LED
  }
}

digitalWrite(ledPin, ledEtat ? HIGH : LOW);
T_previous = temperature;

// Affichage LCD
lcd.setCursor(0, 0);
lcd.print("T:");
lcd.print(temperature, 1);
```

```
  lcd.print("C ");
  lcd.setCursor(10, 0);
  lcd.print((int)percent);
  lcd.print("%   ");
  lcd.setCursor(0, 1);
  if (ledAutoOff) {
    lcd.print("Stable: LED OFF ");
  } else {
    lcd.print("PWM:");
    lcd.print(pwm);
    lcd.print("    ");
  }

  delay(500);
}
```

B. Schéma Proteus



C. Sérial Monitor

XIII. Bibliographie

1. Datasheet **SHT31** – Sensirion.

SHT31-DIS Datasheet(PDF) - Sensirion AG Switzerland

2. Datasheet **BN220 GPS** – Beitian.

GNSS Module BN-220 - 深圳市北天通讯有限公司企业官网

3. Datasheet **IRFZ44R MOSFET**.

IRFZ44R Datasheet(PDF) - International Rectifier

4. Datasheet **Arduino UNO R3** – Arduino.cc.

localhost:8123/A000066/a7uk7l-datasheet.html

5. Rapport de PRT – **INSA Strasbourg (2023/24)**.
6. Ogata K., *Modern Control Engineering*, Pearson, 2010.
7. Bejan A., *Heat Transfer Handbook*, Wiley, 2003.