Système intelligent de protection contre les fuites de gaz combustible

Réalisé par : El fadili Salaheddine
Encadré par : Ahmed Fenkouch
Identifiant :MH031T

PLAN D'ETUDE:

Introduction

1-Description du système

2-Cahier des charges fonctionnelles

3-Problématique

4-Objectifs :

. Objectif 1: Détection de gaz propagé

. Objectif 2: Avertissement de utilisateur

. Objectif 3: extraction de gaz

Conclusion

Introduction :

Figure 1 : l'explosion d’une bonbonne de gaz provoque 13 blessés à Ben M’sik Casablanca le 05/12/2017

Figure 2: explosion d un depot de gaz a Mohemmadia le jeudi 22/12/2022

Description du système

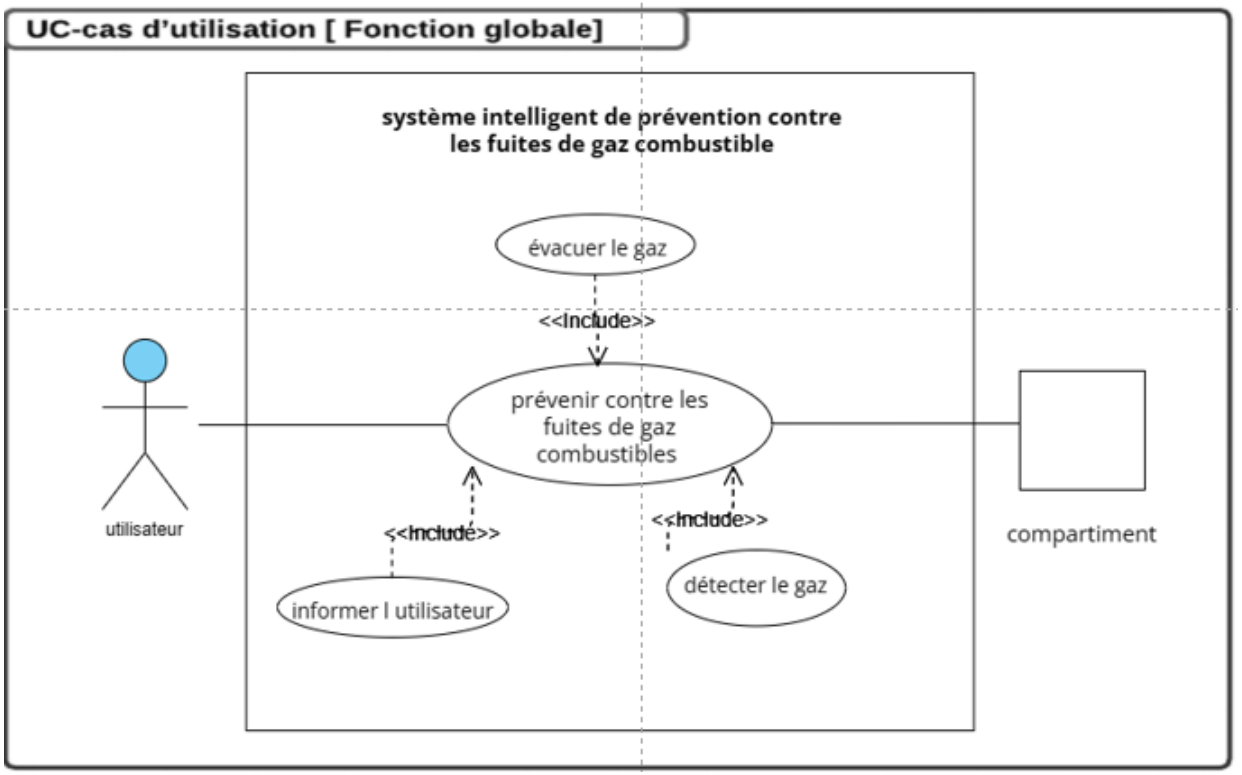


Figure 3 : diagramme des cas d'utilisation

Cahier des charges fonctionnelles :

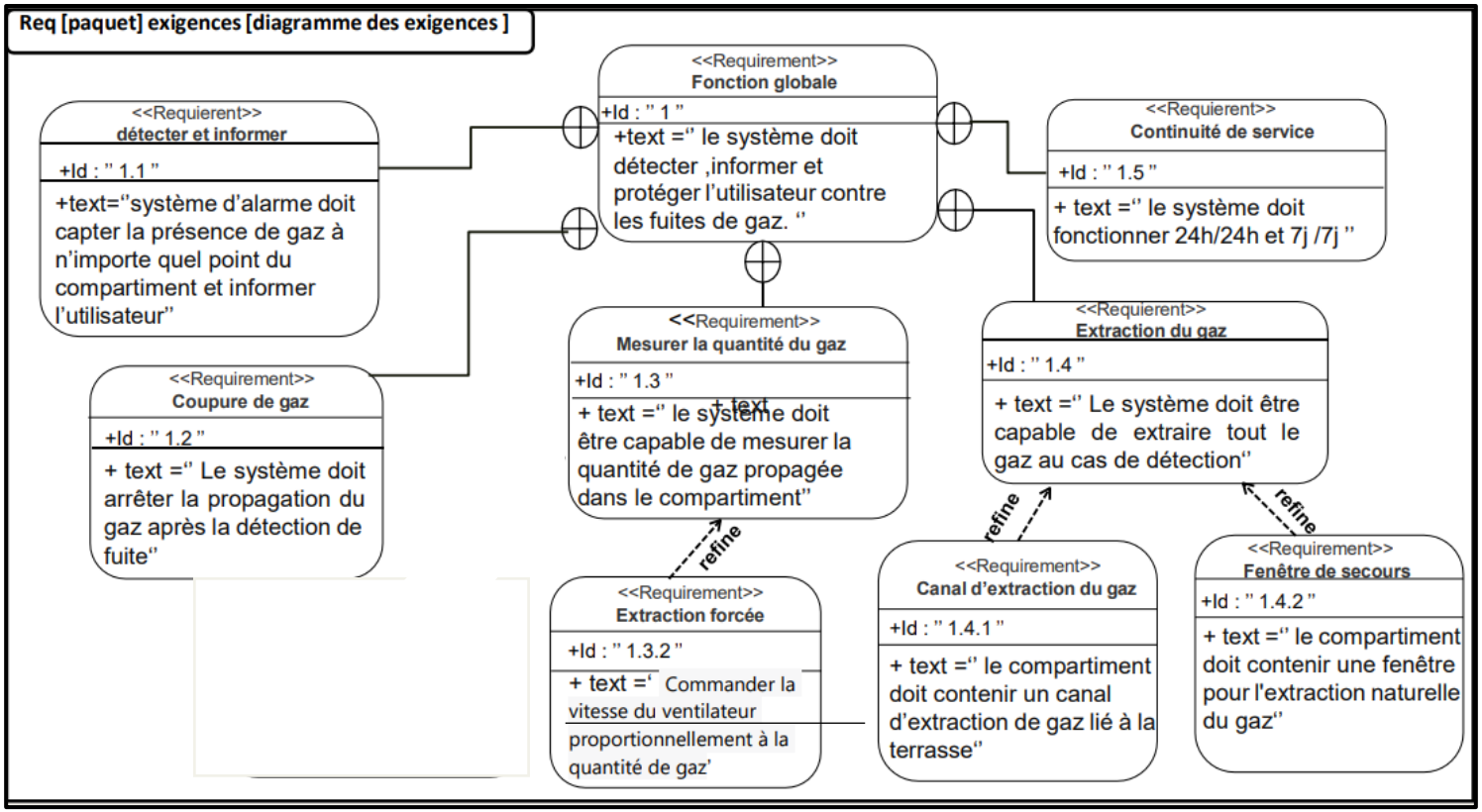


Figure 5 : Diagramme d'exigence

Problématique :

Comment optimiser les performances d'un système de prévention contre les fuites de gaz combustible en évacuant le gaz à l'aide d'un ventilateur extracteur et en alertant l'utilisateur en temps réel à partir de son téléphone portable ?

Objectif 1:

Détection de gaz propagé

- Mise en situation
- Le choix des capteurs
- Principe de fonctionnement
- Détermination de la quantité de gaz propagé

Mise en situation :

- Réaction de combustion :

Combustible + comburant + Energie → (CO2 ou CO) + des produites

- **gaz nature, Butane, l'huile, l'alcool, les matières plastiques... etc.**

- **Dioxygène O2, Ozone O3, Peroxydes, Persulfates, Nitrates ... etc.**

- **étincelle de feu, Haute température, Rayon de soleil... etc.**

Gaz	La limite inférieure d'explosivité	La limite supérieure d'explosivité
Butane	1,8	8,4
Méthane	5,0	15

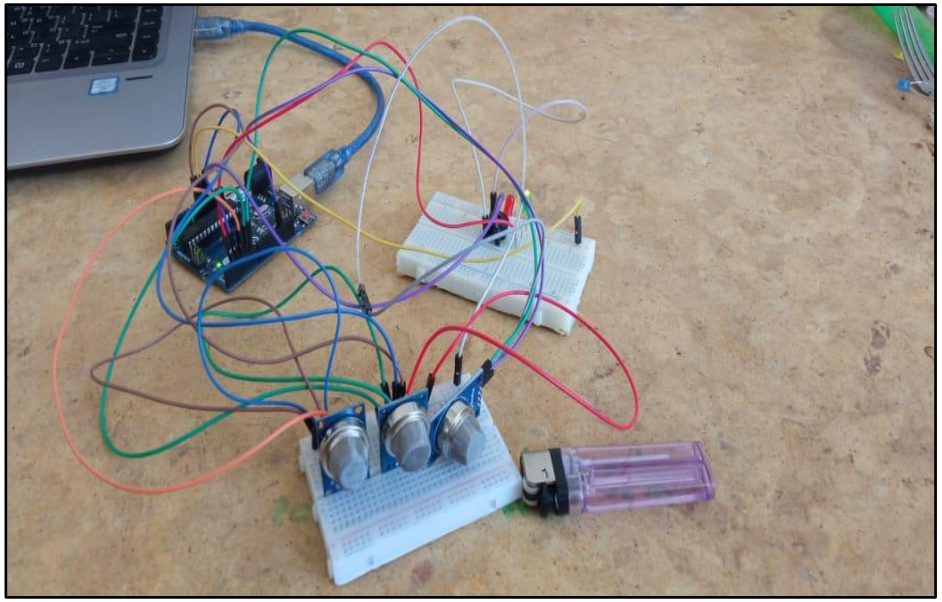
Figure 6 : Les limites explosives du butane et méthane (% par volume d'air)

Le choix des capteurs :

	 Figure : capteur mq-6	 Figure : capteur mq-4	 Figure : capteur mq-2
Tension De Fonctionnement:	+5 V DC	+5 V DC	+5 V DC
Temps de réponse:	3s à 10s	3s à 10s	3s à 10s
Détection	Butane et Propane	Méthane	GPL, l'isobutane, propane, méthane, l'alcool, de l'hydrogène et de la fumée.

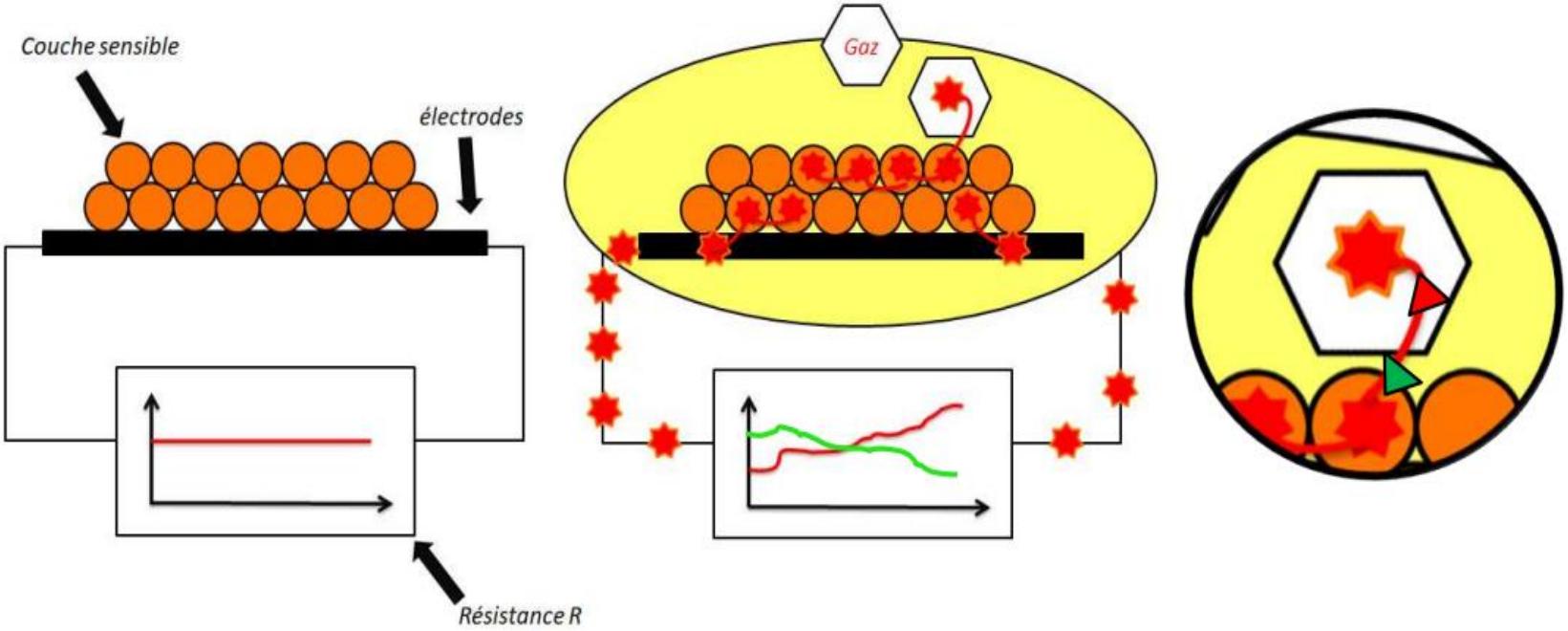
Expérience de la détection de gaz :

```
#define MQ4_D0 5  // Broche 5 (D0) pour la sortie numérique du capteur MQ-4
#define MQ6_D0 4  // Broche 4 (D0) pour la sortie numérique du capteur MQ-6
#define MQ2D_D0 6  // Broche 6 (D0) pour la sortie numérique du capteur MQ-2D
#define LED_GREEN 3  // Broche de sortie pour la LED verte
#define LED_RED 2  // Broche de sortie pour la LED rouge

void setup() {
  pinMode(MQ4_D0, INPUT);  // Configuration de la broche d'entrée comme entrée pour le capteur MQ-4
  pinMode(MQ6_D0, INPUT);  // Configuration de la broche d'entrée comme entrée pour le capteur MQ-6
  pinMode(MQ2D_D0, INPUT);  // Configuration de la broche d'entrée comme entrée pour le capteur MQ-2D
  pinMode(LED_GREEN, OUTPUT);  // Configuration de la broche de sortie pour la LED verte
  pinMode(LED_RED, OUTPUT);  // Configuration de la broche de sortie pour la LED rouge
}

void loop() {
  int gasDetected = digitalRead(MQ4_D0) || digitalRead(MQ6_D0) || digitalRead(MQ2D_D0);
  // Utilisation de l'opérateur logique OR pour vérifier si l'un des capteurs détecte du gaz

  if (gasDetected == HIGH) {
    digitalWrite(LED_GREEN, LOW);  // Éteindre la LED verte
    digitalWrite(LED_RED, HIGH);  // Allumer la LED rouge
  } else {
    digitalWrite(LED_GREEN, HIGH);  // Allumer la LED verte
    digitalWrite(LED_RED, LOW);  // Éteindre la LED rouge
  }

  delay(1000);  // Attente d'une seconde avant de lire à nouveau l'état des sorties numériques des capteurs
}
```



Principe de fonctionnement :

Les capteurs MQ-6, MQ-4 et MQ-2 sont basés sur la technologie de détection de gaz par conductivité des semi-conducteurs. Voici comment ces capteurs fonctionnent de manière générale :



Etude detaillé du capteur MQ2

- Structure interne du capteur M2 :

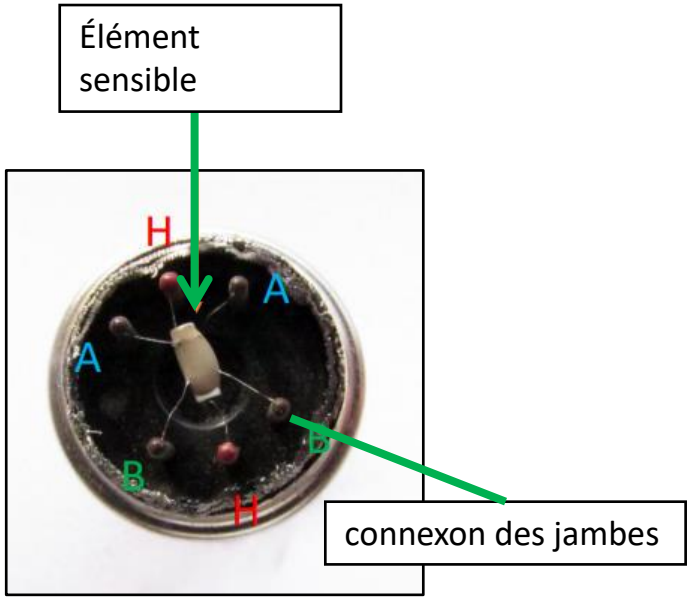


Figure 7 : La structure intérieure de MQ-2

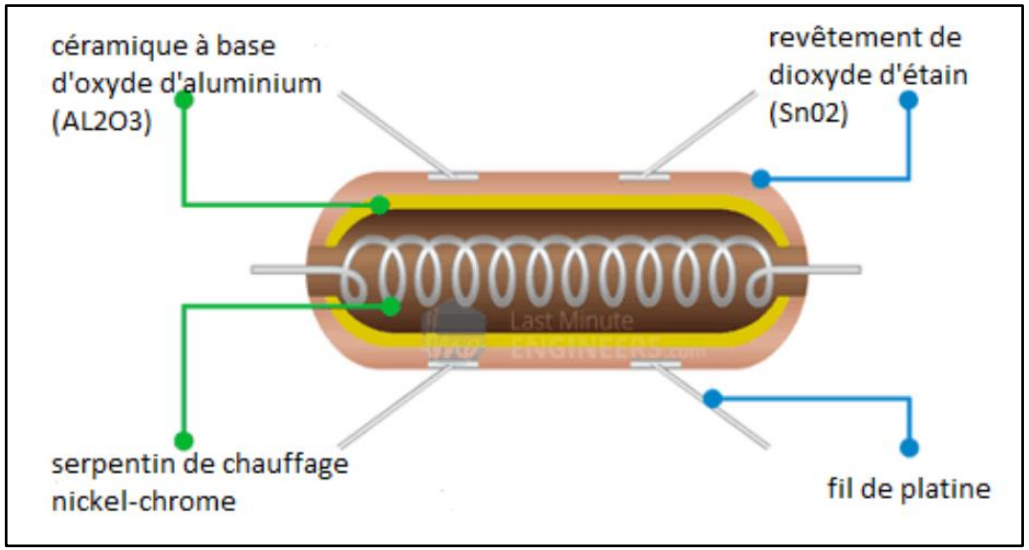


Figure 8 : Structure de l'élément de détection

- Structure interne du capteur M2 :

Le capteur MQ2

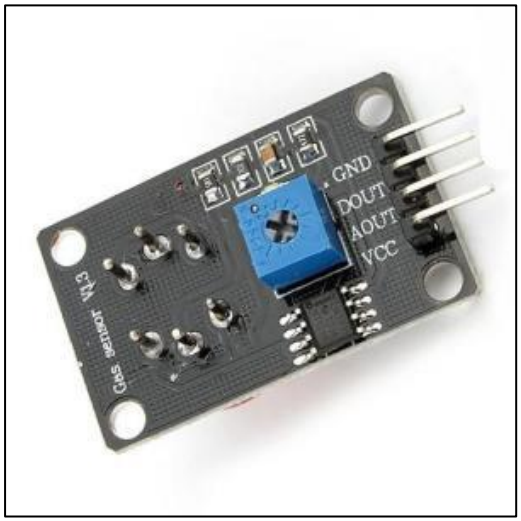


Figure 9 : face arrière du capteur

Shield du capteur

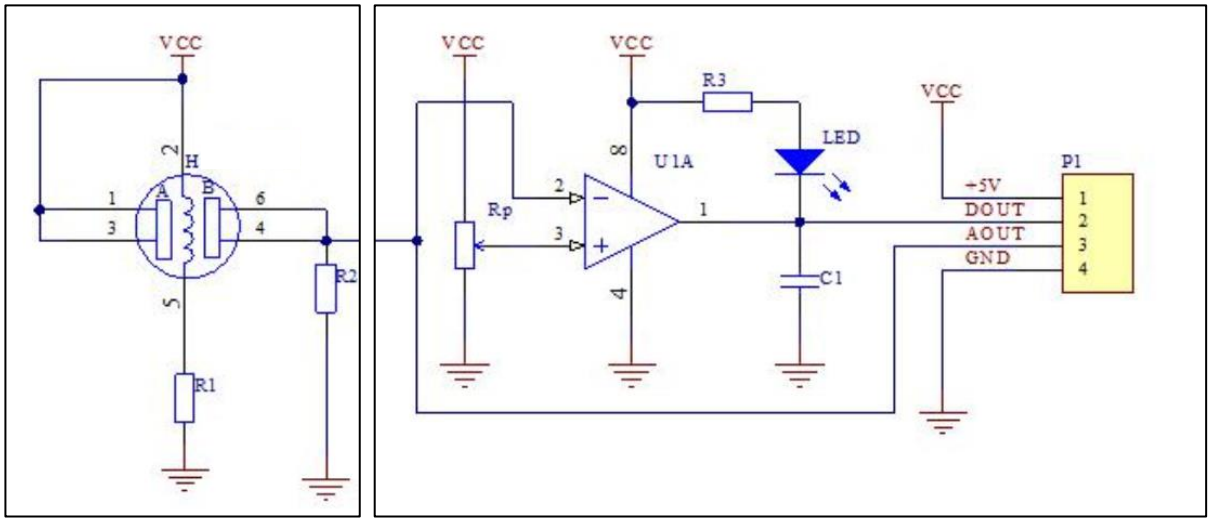


Figure 10 : Schéma électrique du module MQ2

Détermination de la quantité de gaz propagé :

la relation entre X et Y peut être approximée par une équation linéaire de la forme : $Y - y1 = m(X - x1)$

où y1 et x1 sont les coordonnées d'un point sur la ligne droite et m est la pente de la ligne droite.

Ce qui implique : $X = (Rs/Ro - y1)/m + x1$

D'où : $\log X = (\log (Rs/Ro) - y1)/m + x1$

Et enfin :

ppm=X = 10^((log (Rs/Ro) - y1) / m +x1).

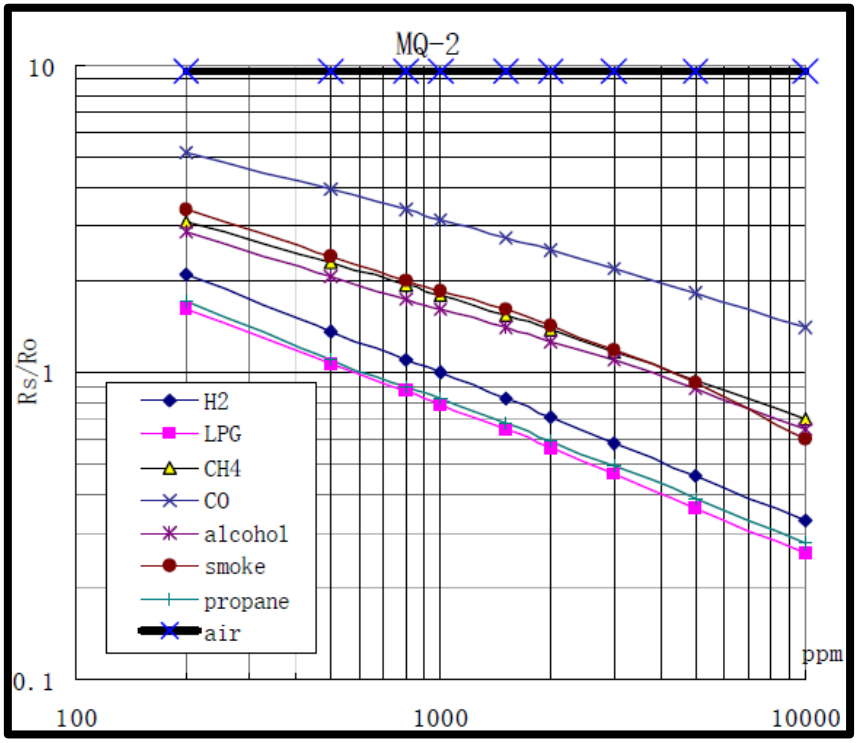
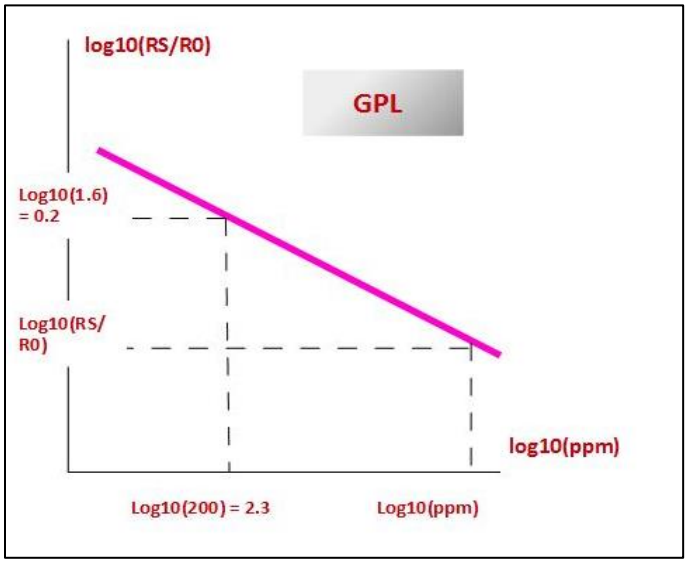


Figure 11 :Courbe du ratio RS/R0 en fonction de la concentration en gaz en ppm(MQ-2 Datasheet)

Détermination de la quantité de gaz propagé (Gaz de Pétrole Liquéfié) :



pente = -0,45 = (log10 (RS/R0) – 0.2) / (log10(ppm) -2.3)

log10 (ppm) = 2.3 + (log10 (RS/R0) – 0.2) / – 0.45

ppm = 10 ^ (2.3 + (log10 (RS/R0) – 0.2) / – 0.45)

```
// Déclaration des constantes
const int gasSensorPin = A0;  // Broche analogique utilisée pour lire les valeurs du capteur
const float R0 = 9.83;        // Valeur de résistance de référence à l'air propre

// Configuration initiale
void setup() {
  Serial.begin(9600);  // Initialisation de la communication série
}

// Boucle principale
void loop() {
  // Lecture de la valeur analogique
  int sensorValue = analogRead(gasSensorPin);

  // Calcul de la résistance du capteur (RS)
  float voltage = sensorValue * (5.0 / 1023.0);  // Conversion de la valeur analogique en tension
  float RS = ((5.0 - voltage) / voltage) * R0;

  // Calcul de la concentration de gaz en ppm
  float ppm = pow(10, (2.3 + (log10(RS / R0) - 0.2) / -0.45));

  // Affichage de la valeur en ppm
  Serial.print("Concentration de gaz GPL en ppm : ");
  Serial.println(ppm);

  delay(1000);  // Attendre 1 seconde avant la prochaine lecture
}
```

Test du capteur :

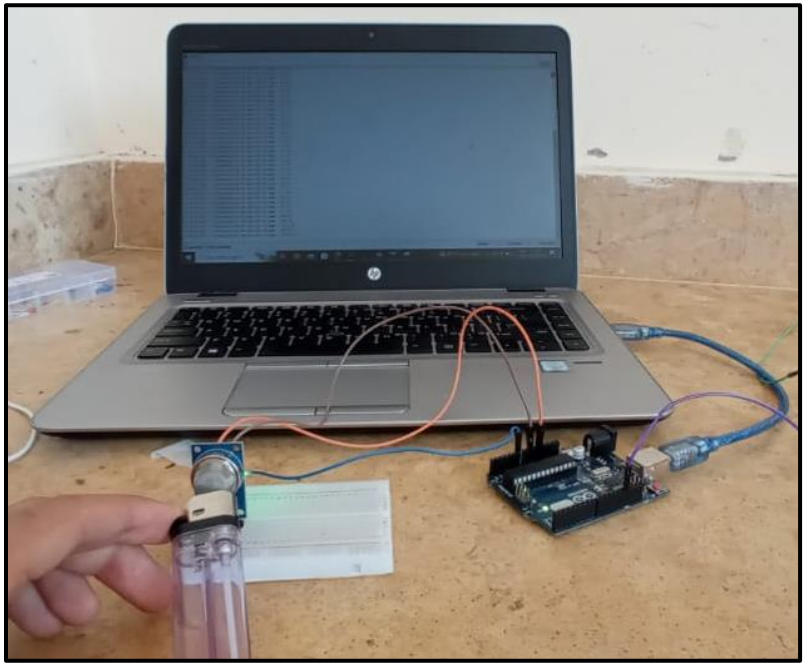
COM6

Send

```
17:28:16.897 -> Concentration de gaz GPL en ppm : 37.39
17:28:17.866 -> Concentration de gaz GPL en ppm : 40.30
17:28:18.866 -> Concentration de gaz GPL en ppm : 37.39
17:28:19.897 -> Concentration de gaz GPL en ppm : 269.23
17:28:20.896 -> Concentration de gaz GPL en ppm : 864.70
17:28:21.881 -> Concentration de gaz GPL en ppm : 592.51
17:28:22.881 -> Concentration de gaz GPL en ppm : 429.36
17:28:23.911 -> Concentration de gaz GPL en ppm : 903.49
17:28:24.896 -> Concentration de gaz GPL en ppm : 577.64
17:28:25.880 -> Concentration de gaz GPL en ppm : 359.76
17:28:26.910 -> Concentration de gaz GPL en ppm : 242.56
17:28:27.895 -> Concentration de gaz GPL en ppm : 454.28
17:28:28.926 -> Concentration de gaz GPL en ppm : 287.97
17:28:29.910 -> Concentration de gaz GPL en ppm : 1938.56
```

Autoscroll Show timestamp Newline 9600 baud Clear output

```
  // Calcul de la résistance du capteur (RS)
  float voltage = sensorValue * (5.0 / 1023.0);  // Conversion de la valeur analogique en tension
  float RS = ((5.0 - voltage) / voltage);

  // Calcul de la concentration de gaz en ppm
  float ppm = pow(10, (2.3 + (log10(RS / R0) - 0.2) / -0.45));

  // Affichage de la valeur en ppm
  Serial.print("Concentration de gaz GPL en ppm : ");
  Serial.println(ppm);

  delay(1000);  // Attendre 1 seconde avant la prochaine lecture
}
```



Objectif 2:

Avertissement de utilisateur

- **Representation des composants**
- **Avertissement visuel , auditive et via application mobile**

Representation des composants:

Figure 12 : LED Rouge et Verte

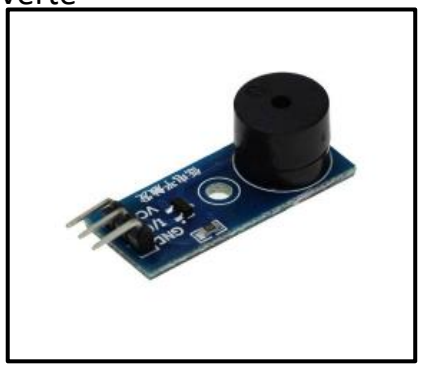


Figure 13 : Buzzer

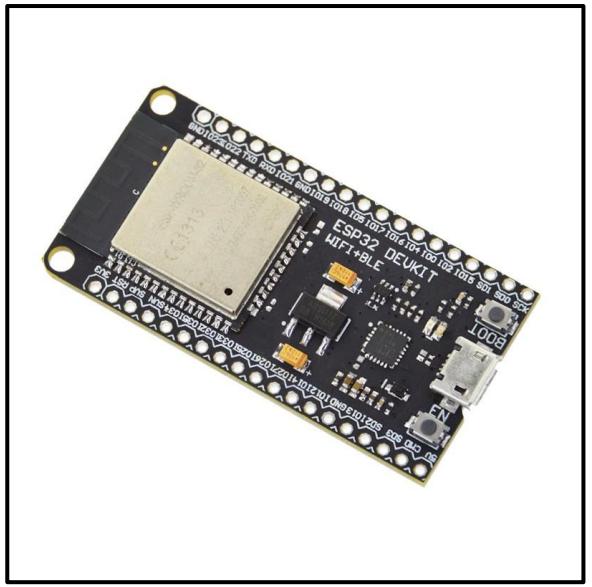


Figure 14 : NodeMCU ESP32

Avertissement via application mobile

1-Partie hardware

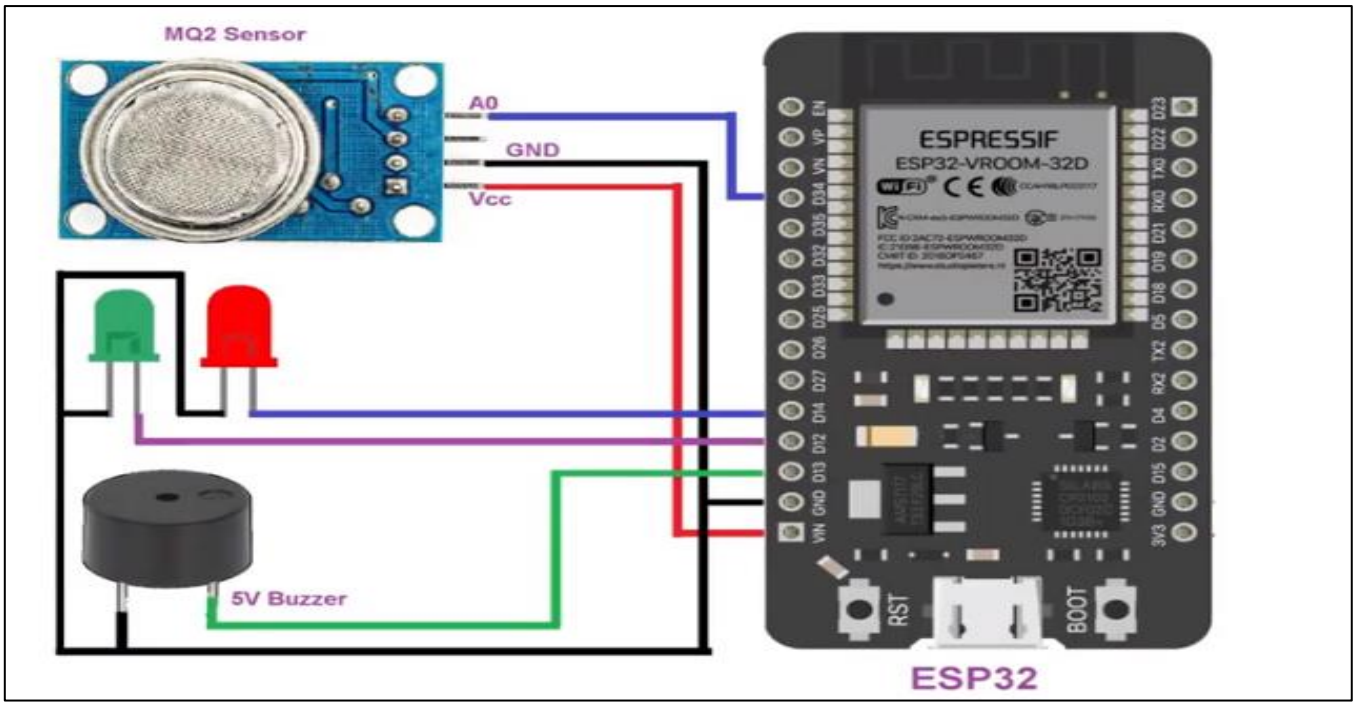


Figure 15 : Câblage du Buzzer , led et ESP32

Avertissement via application mobile

2-Partie Software

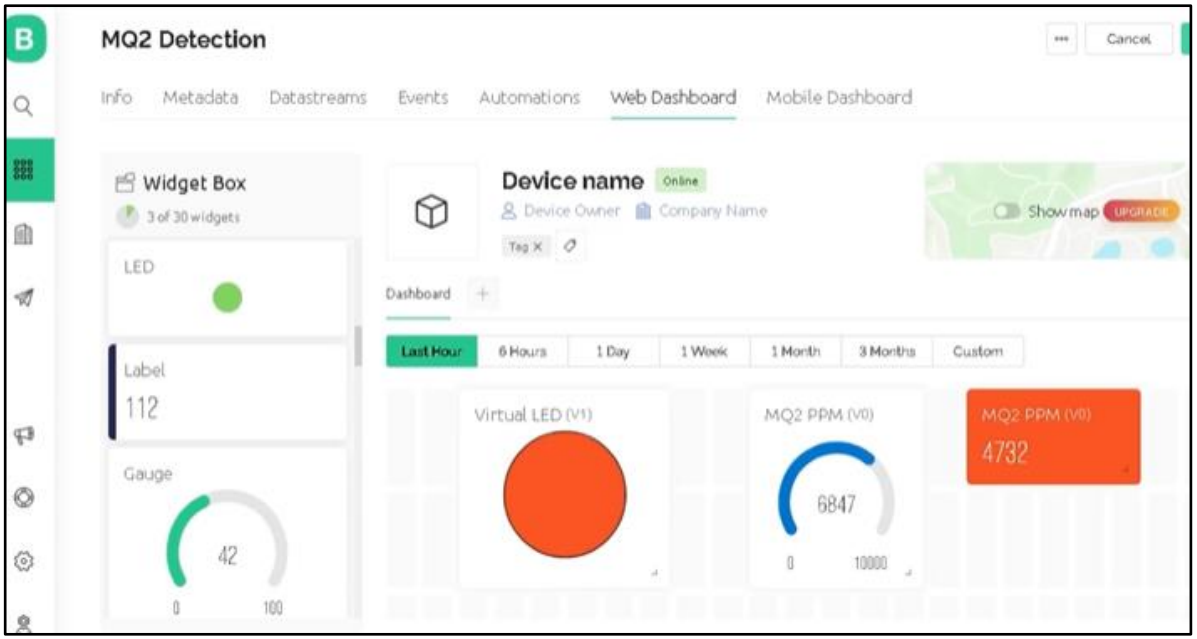
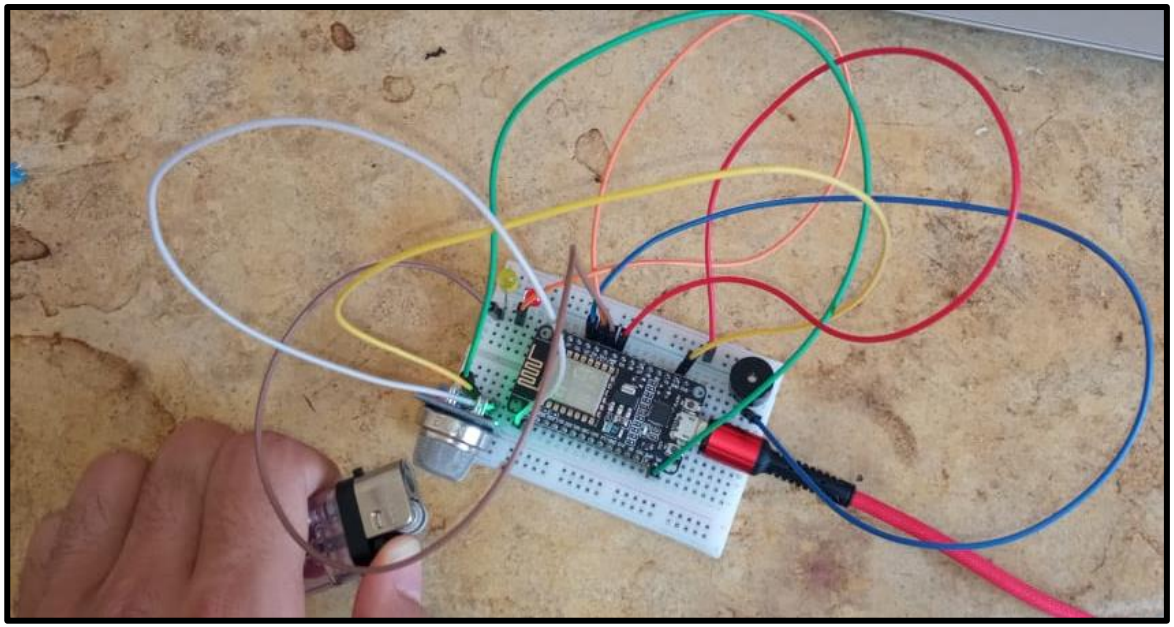
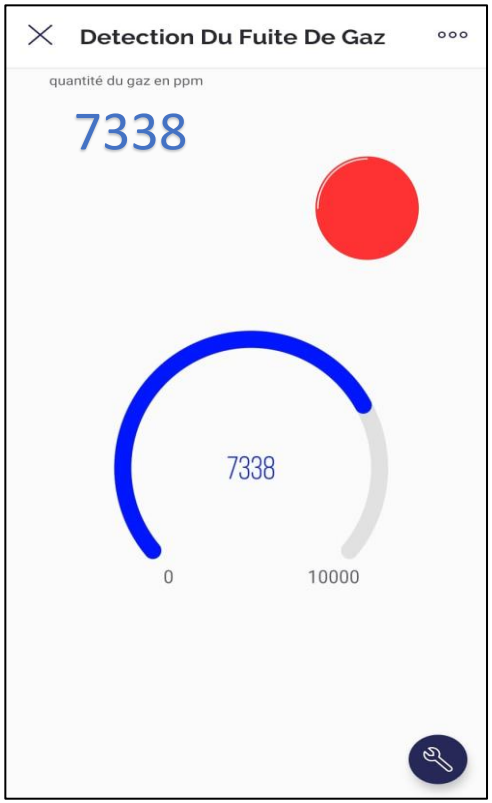


Figure 16 : création interface utilisateur pour contrôler les fonctionnalités de notre ESP32

Réalisation du système d Avertissement





Objectif 3:

Evacuation du Gaz

- **Couper le flux de gaz**
- **Commande du ventilateur**
- **Régulation du quantité du gaz**

Couper le flux de gaz :

Figure 17: Electrovanne

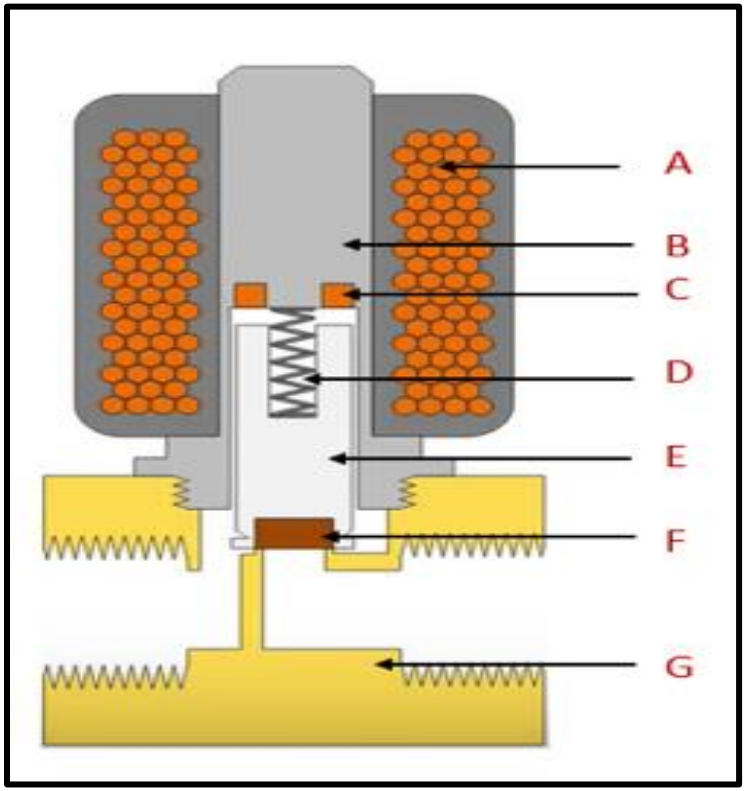


Figure 18 : Composants d'une électrovanne ; bobine (A) ; induit (B) ; bague de déphasage (C) ; ressort (D) ; plongeur (E) ; joint (F) ; corps de l'électrovanne (G)

Evacuation du gaz

La vitesse du ventilateur doit être proportionnelle à la quantité de gaz détectée, ce qui permettra de réguler la quantité de gaz dans une chambre et de maintenir sa valeur minimale

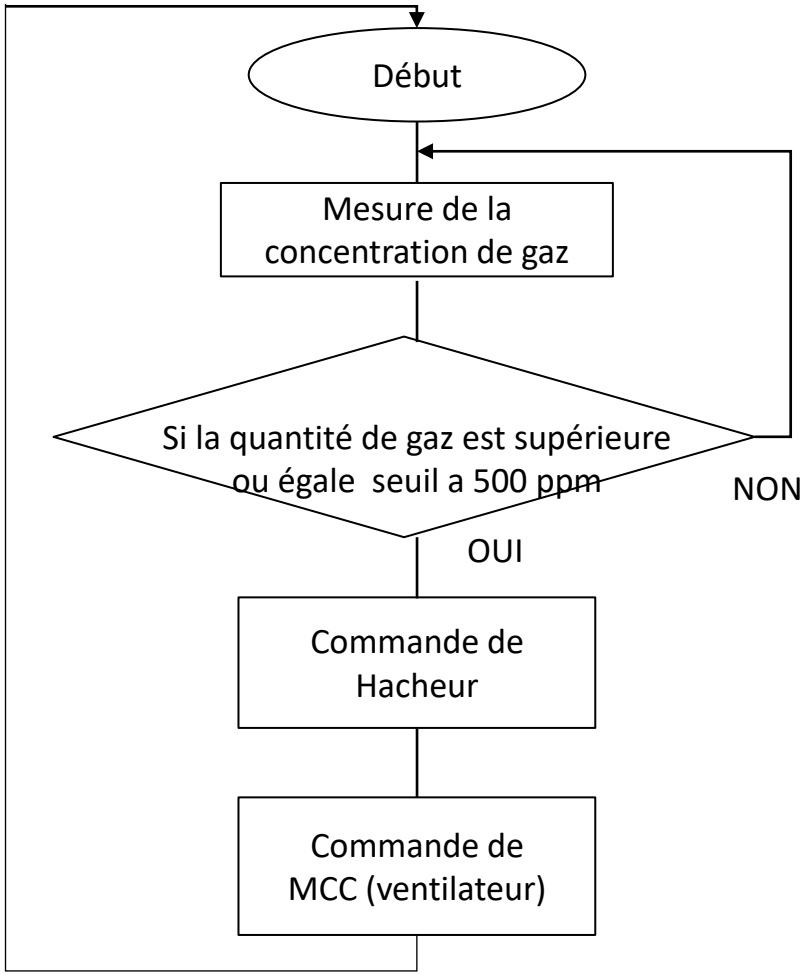


Figure 19:Organigramme décrivant la motoristion du mcc

Commande du ventilateur à laide du transistor :

Caractéristiques du transistor IRFZ44N

- MOSFET à canal N petit signal
- Le courant de drainage continu (DI) est de 49 A à 25 ° C
- La tension de seuil de porte minimale (VGS-th) est de 2 V
- La tension de seuil maximale de la porte (VGS-th) est de 4 V
- La tension Gate-Source est (VGS) est ± 20V (max)
- La tension drain-source maximale (VDS) est de 55V

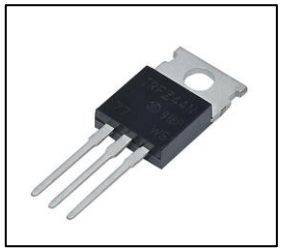


Figure 19 : transistor IRFZ44N

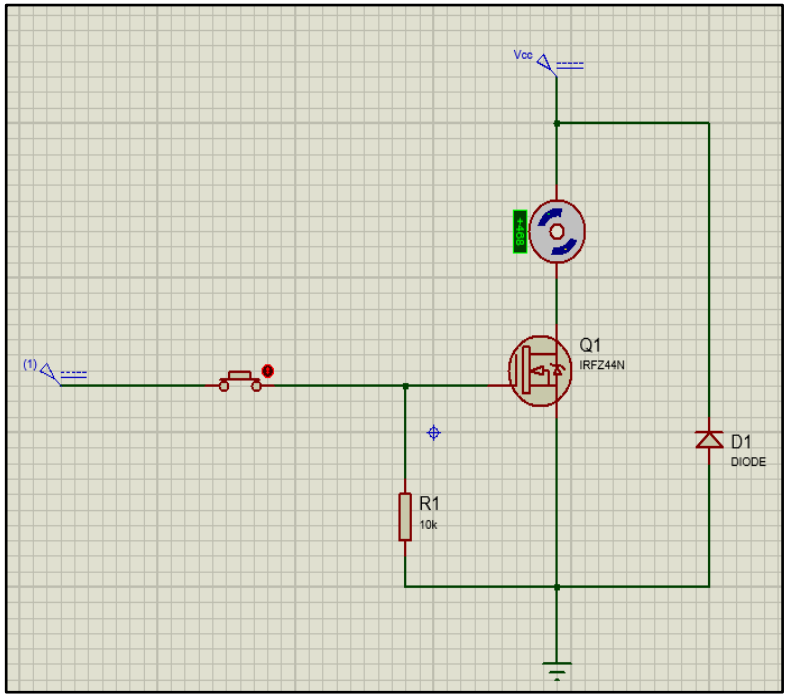


Figure20: Commande du MCC par le transistor IRFZ44N

Hypotheses de simplification:

1-Si la quantité du gaz est compris entre 500 ppm et 5000 ppm :

- La relation est linéaire entre la quantité du gaz et la vitesse
- La vitesse maximale du ventilateur est atteinte lorsque la quantité de gaz est de 5 000 ppm au plus.
- la vitesse est minimale (19.6% de Vmax) est atteinte lorsque la quantité de gaz est égale a 500 ppm

Niveau	Feu indicateur	LPG (ppm)
Securite	Green	<500
précautio n	Yellow	> = 500 < = 1000
Dangeureux	Red	> 1000

Figure 21 : GPLppm niveau

2-Si la quantité du gaz est inférieure a 500 ppm , la vitesse est nulle.

3-Si la quantité du gaz dépasse 5000 ppm , la vitesse est maximale.

Commande du Transistor par Arduino Uno:

Calcul de la valeur MLI (PWM générée par Arduino):

1-Si la quantité du gaz est inferieure a 500 ppm , PWM=0

2-Si la quantité du gaz dépasse 5000 ppm , PWM=255

3-Si la quantité du gaz est compris entre 500 ppm et 5000 ppm :

on sait que :
50 = a * 500 + b (50 valeur liee a 19.6% de Vmax)
255 = a * 5000 + b
Alors :
a= 0.045
B= 47

On trouve :
PWM = 0.0455556 × ppm + 47.0556.

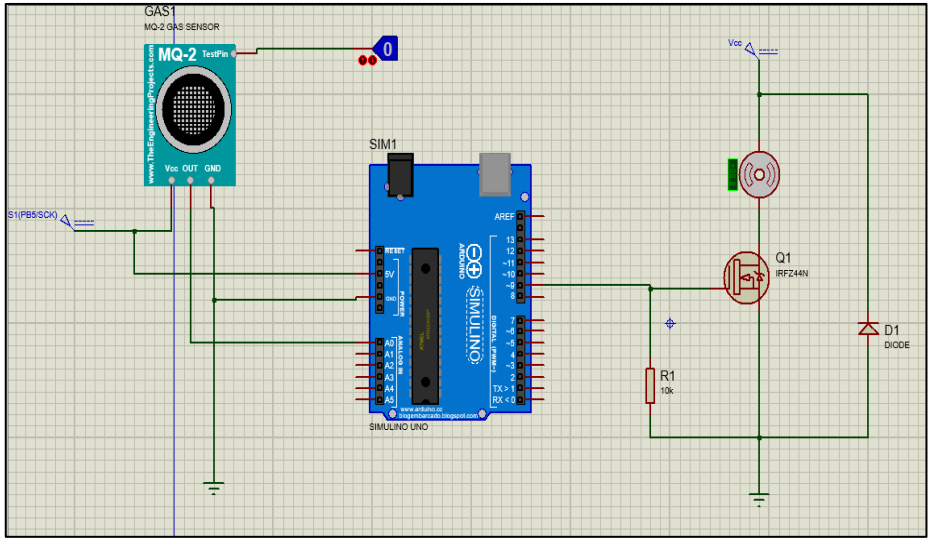


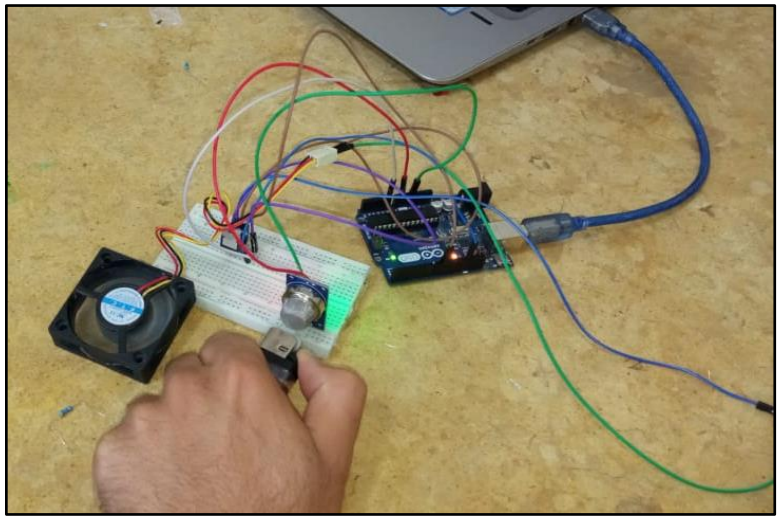
Figure22: Commande du transistor par Arduino

processus de régulation de la quantité de gaz en dessous de 500 ppm

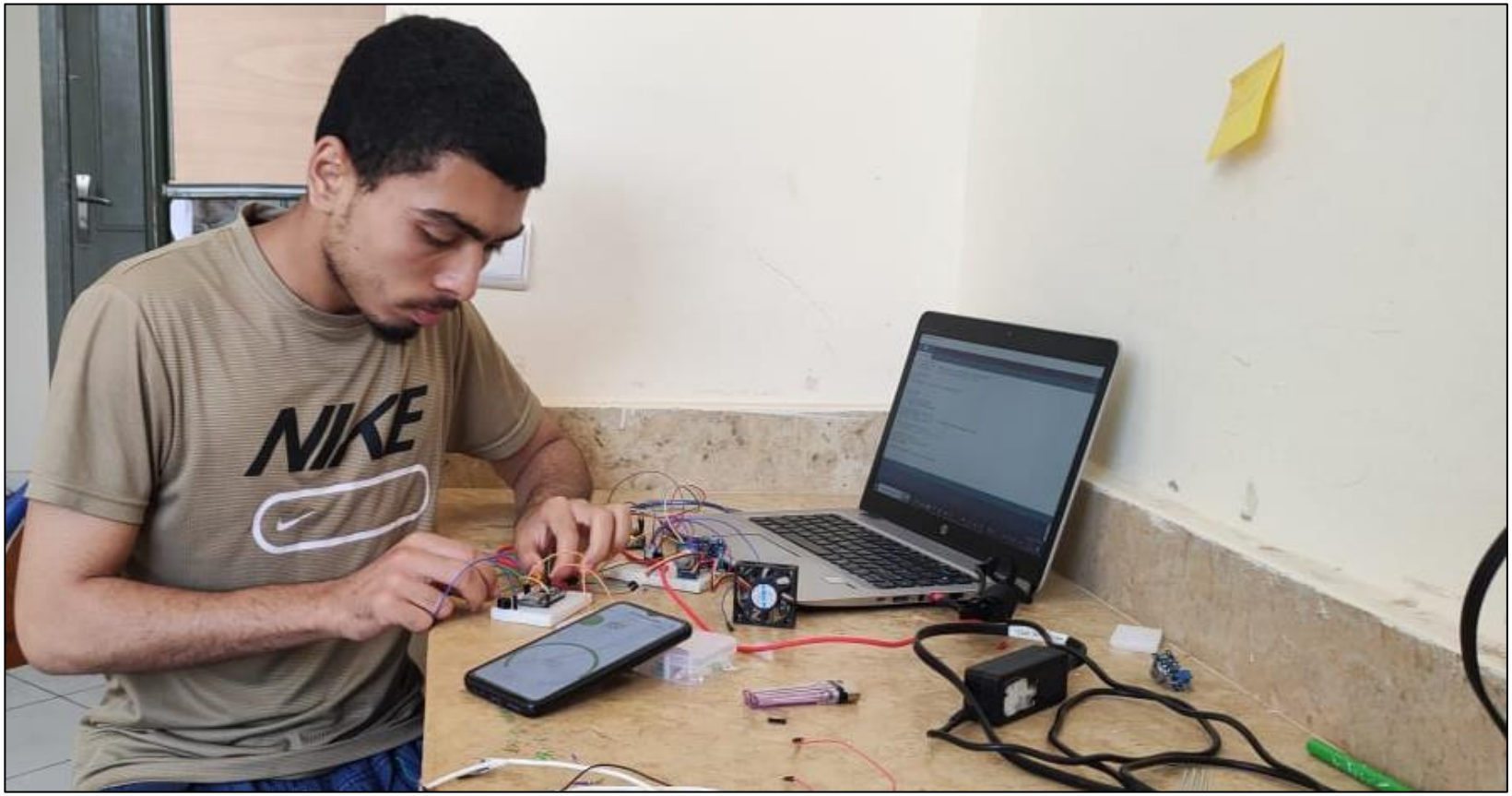
Début
Tant que vrai (boucle infinie)
1. Calculer la concentration de gaz en ppm.
2. Si ppm < 500 alors
. Régler PWM à 0 (arrêter le ventilateur).
3. Sinon, si ppm >= 500 et ppm <= 5000 alors
8. Calculer PWM en utilisant la relation PWM = 0.0455556 * ppm + 47.0556.
4. Sinon (ppm >= 5000)
. Régler PWM à 255 (pleine vitesse du ventilateur).
5. Régler la vitesse du ventilateur en utilisant la fonction analogWrite() avec la valeur PWM.
6. Répéter après un certain délai (par exemple, 1 seconde).
Fin Tant que
Fin

Test du commande du ventilateur :

```
// Déclaration des constantes
const int gasSensorPin = A0;    // Broche analogique utilisée pour lire les valeurs du capteur
const float R0 = 9.83;          // Valeur de résistance de référence à l'air propre
const int pinVentilateur = 9;   // Broche de commande du ventilateur

void setup() {
  pinMode(pinVentilateur, OUTPUT);   // Configuration de la broche du ventilateur en mode de sortie
  Serial.begin(9600);                // Initialisation de la communication série
}

void loop() {
  // Lecture de la valeur analogique du capteur MQ-2
  int sensorValue = analogRead(gasSensorPin);

  // Calcul de la résistance du capteur (RS)
  float voltage = sensorValue * (5.0 / 1023.0);  // Conversion de la valeur analogique en tension
  float RS = R0 * ((5.0 - voltage) / voltage);

  // Calcul de la concentration de gaz en ppm
  float ppm = pow(10, (2.3 + (log10(RS / R0) - 0.2) / -0.45));

  // Régulation de la vitesse du ventilateur en fonction de la concentration de gaz
  int pwm = 0;

  if (ppm < 500) {
    pwm = 0;
  } else if (ppm >= 500 && ppm <= 5000) {
    pwm = 0.0455556 * ppm + 47.0556;
  } else {
    pwm = 255;
  }

  // Commande du ventilateur avec la valeur PWM calculée
  analogWrite(pinVentilateur, pwm);
```



Réalisation d'un prototype :



CONCLUSION GÉNÉRALE

Merci
pour
votre
attention