

# Projet Ventilateur Grow Box

Dans ce projet on cherche à répondre à un problème qui est la surchauffe de panneau LED dans un Grow Box à l'espace Greenlab. Pour y remédier, le but est de créer un programme sur un Arduino UNO qui va permettre d'activer un ventilateur lorsque la température est trop élevée. Pour l'instant j'ai un schéma de montage avec une carte Arduino UNO branché à un capteur de température qui, en fonction de la température, va ouvrir ou fermer un relai ce qui aura pour effet d'ouvrir ou de fermer le circuit du ventilateur. Le code que j'ai utilisé est le suivant :

```
#include <Adafruit_Sensor.h>
#include <DHT.h>

#define DHTpin 2      // pin2 devient le pin du DHT11
#define Relai 3       // pin3 devient le pin du relai
#define DHTTYPE DHT22 // on règle le modèle de capteur
DHT dht(DHTpin, DHTTYPE);

void setup() {
  pinMode(DHTpin, INPUT);    // règle le pin 2 en input
  pinMode(Relai, OUTPUT);    // règle le pin 3 en output
  Serial.begin(9600);        // initialise la comm.
  dht.begin();               // allume le capteur
}

void loop() {

  delay(2000);               //attend un peu entre chaque mesures

  float t = dht.readTemperature(); // lit la température exterieur et l'associe à la variable t=température

  if ( t >= 25 ) {           // si la temérature est au dessus de 35°C
    digitalWrite(Relai, HIGH); // le relai se ferme => le ventilateur s'allume
    Serial.println("closed");
  }
```

```

else {
    // la température est a 35°C ou moins
    digitalWrite(Relai, LOW);    // le relai s'ouvre => le ventilateur s'arrête
    Serial.println("open");
}
if (isnan(t)) {
    Serial.println("x(");    // verifie le bon fonctionnement du capteur
}
Serial.println(t);
}

```

Maintenant, il ne reste plus qu'à rajouter des fonctionnalités tels qu'un écran LCD avec possibilité de régler la température de déclenchement, mais aussi des moyens de régler l'hygrométrie !!

J'ai ajouté ci-joint les librairies utilisé dans ce code.

Paul SPIRCKEL : J'ai ajouté un LCD (<https://learn.adafruit.com/character-lcds/wiring-a-character-lcd>). A chaque déclenchement du relais, il est parasité par de nombreux de caractères mais le circuit fonctionne toujours en arrière-plan.

A noter qu'il faudrait ajouter un petit delta de température de façon à ce que le ventilateur ne s'active/désactive pas sans arrêt autour de 25°C (typiquement ventiler jusqu'à 22°C avant de se désactiver).

Voici le code mis à jour :

```

#include <Adafruit_Sensor.h>

#include <DHT.h>

#include <LiquidCrystal.h>

#define DHTpin 2    // pin2 devient le pin du DHT11

#define Relai 3    // pin3 devient le pin du relai

#define DHTTYPE DHT22    // on règle le modèle de capteur

DHT dht(DHTpin, DHTTYPE);

LiquidCrystal lcd(7, 8, 9, 10, 11, 12);    // connexion des ports de l'écran LCD

```

```
void setup() {

    pinMode(DHTpin, INPUT);    // règle le pin 2 en input

    pinMode(Relai, OUTPUT);    // règle le pin 3 en output

    Serial.begin(9600);        // initialise la comm.

    dht.begin();               // allume le capteur

    lcd.begin(16, 2);

    lcd.setCursor(1, 0);

    lcd.write("Temperature =");

    lcd.setCursor(6, 1);

    lcd.write(" C");

}

void loop() {

    delay(2000);                //attend un peu entre chaque mesures

    float t = dht.readTemperature(); // lit la température exterieur et l'associe à la variable t=température

    lcd.setCursor(0, 1);

    lcd.print(dht.readTemperature());
```

```

if ( t <= 25 ) {          // si la température est au dessus de 35°C

    digitalWrite(Relai, HIGH);    // le relai se ferme => le ventilateur s'allume

    Serial.println("closed");

}

else {                    // la température est a 35°C ou moins

    digitalWrite(Relai, LOW);     // le relai s'ouvre => le ventilateur s'arrête

    Serial.println("open");

}

if (isnan(t)) {

    Serial.println("x(");        // verifie le bon fonctionnement du capteur

}

Serial.println(t);

}

```

Paul SPIRCKEL : J'ai finalisé le code pour 1 ventilateur, avec un delta de température de 3°C et un potentiomètre permettant de choisir le seuil de température à ne pas dépasser. Le LCD ne fonctionne toujours pas correctement. Il peut s'agir d'un problème matériel donc je n'ai rien changé tant que le circuit ne sera pas posé sur une platine. Aussi, il faut faire attention à ne pas allumer le circuit avec le potentiomètre réglé pour une température en dessous de la température ambiante sinon le ventilateur ne s'arrêtera jamais de tourner. Si c'est le cas, il faut juste éteindre le circuit, remonté le potentiomètre et tout rallumer.

```

#include <Adafruit_Sensor.h>

#include <DHT.h>

```

```
#include <LiquidCrystal.h>

#define DHTpin 2          // pin2 devient le pin du DHT11

#define Relai 3           // pin3 devient le pin du relai

#define DHTTYPE DHT22     // on règle le modèle de capteur

DHT dht(DHTpin, DHTTYPE);

LiquidCrystal lcd(7, 8, 9, 10, 11, 12); // connexion des ports de l'écran LCD

int adcPin = A0;          // attribution du pin analogique A0 comme entrée du signal du potentiomètre
int poten = 0;            // poten est la variable modifiée par le potentiomètre

void setup() {

    pinMode(DHTpin, INPUT); // règle le pin 2 en input

    pinMode(Relai, OUTPUT);  // règle le pin 3 en output

    pinMode(adcPin, INPUT);  // règle le pin A0 (analogique) comme entrée du signal du potentiomètre

    Serial.begin(9600);      // initialise la comm.

    dht.begin();             // allume le capteur

    lcd.begin(16, 2);

    lcd.setCursor(0, 0);

    lcd.println("Temp =");

    lcd.setCursor(0, 1);

    lcd.println("Set = ");

}
```

```

void loop() {

    delay(2000);           //attend un peu entre chaque mesures

    poten = (analogRead(adcPin)/10); // poten prend comme valeur celle envoyée par le potentiomètre, divisée
    par 10 pour avoir un interval de température entre 0 et 70°C

    Serial.println(3+poten);      // on fixe une valeur minimum de T à 3°C de façon à créer un delta en
    additionnant la valeur poten

    lcd.setCursor(6, 1);

    lcd.println(3+poten);

    if ( dht.readTemperature() >= (3+poten) ) {           // si la température est au dessus de (3+poten)

        while (dht.readTemperature() >= (poten)) {       // et tant qu'elle n'est pas repassée en dessous de
        poten (donc 3°C en dessous de la valeur de déclenchement), on active le ventilateur

            delay(2000);

            digitalWrite(Relai, LOW);    // le relai se ferme => le ventilateur s'allume

            Serial.println(dht.readTemperature());

            Serial.println("closed");

            lcd.setCursor(7, 0);

            lcd.write(dht.readTemperature());

        }

    }

    else {           // la température est a 35°C ou moins

        digitalWrite(Relai, HIGH);      // le relai s'ouvre => le ventilateur s'arrête
    }
}

```

```
Serial.println(dht.readTemperature());

Serial.println("open");

lcd.setCursor(7, 0);

lcd.write(dht.readTemperature());

}

}
```

## Avec 2 relais

```
#include <Adafruit_Sensor.h>

#include <DHT.h>

#include <LiquidCrystal.h>

#define DHTpin 2          // pin2 devient le pin du DHT11

#define Relai_1 3         // pin3 devient le pin du relai 1

#define Relai_2 4         // pin4 devient le pin du relai 2

#define DHTTYPE DHT22     // on règle le modèle de capteur

DHT dht(DHTpin, DHTTYPE);

LiquidCrystal lcd(7, 8, 9, 10, 11, 12); // connexion des ports de l'écran LCD

int adcPin = A0;          // attribution du pin analogique A0 comme entrée du signal du potentiomètre
int poten = 0;            // poten est la variable modifiée par le potentiomètre

void setup() {

    pinMode(DHTpin, INPUT); // règle le pin 2 en input
```

```
pinMode(Relai_1, OUTPUT);    // règle le pin 3 en output

pinMode(Relai_2, OUTPUT);    // règle le pin 4 en output

pinMode(adcPin, INPUT);      // règle le pin A0 (analogique) comme entrée du signal du potentiomètre

Serial.begin(9600);          // initialise la comm.

dht.begin();                 // allume le capteur

lcd.begin(16, 2);

lcd.setCursor(0, 0);

lcd.println("Temp =");

lcd.setCursor(0, 1);

lcd.println("Set = ");

}

void loop() {

    delay(2000);              //attend un peu entre chaque mesures

    poten = (analogRead(adcPin)/10); // poten prend comme valeur celle envoyée par le potentiomètre, divisée
    par 10 pour avoir un interval de température entre 0 et 70°C

    Serial.println(3+poten);    // on fixe une valeur minimum de T à 3°C de façon à créer un delta en
    additionnant la valeur poten

    lcd.setCursor(6, 1);

    lcd.println(3+poten);

    if ( dht.readTemperature() >= (3+poten) ) {                // si la température est au dessus de (3+poten)
```



```

while (dht.readTemperature() >= (poten)) {           // et tant qu'elle n'est pas repassée en dessous de
poten (donc 3°C en dessous de la valeur de déclenchement), on active le ventilateur

    delay(2000);

    digitalWrite(Relai_1, LOW);    // le relai se ferme => le ventilateur s'allume

    Serial.println(dht.readTemperature());

    Serial.println("closed 1");

    lcd.setCursor(7, 0);

    lcd.write(dht.readTemperature());

    while (dht.readTemperature() >= (poten+2)) {       // et tant qu'elle n'est pas repassée en dessous
de poten (donc 3°C en dessous de la valeur de déclenchement), on active le ventilateur

        delay(2000);

        digitalWrite(Relai_2, LOW);    // le relai se ferme => le ventilateur s'allume

        Serial.println(dht.readTemperature());

        Serial.println("closed 2");

    }

}

}

}

else {           // la température est a 35°C ou moins

    digitalWrite(Relai_1, HIGH);    // le relai s'ouvre => le ventilateur s'arrête

    Serial.println(dht.readTemperature());

    Serial.println("open");

```

```
lcd.setCursor(7, 0);

lcd.write(dht.readTemperature());

}

}
```

Miro Von der Borch : J'ai aujourd'hui tenté de résoudre certains problèmes de la version précédente du code et du montage :

- Des bugs d'affichage sur l'écran LDC

- La communication serial qui ne renvoie rien

- La valeur des potentiomètres qui se fige si elle est en dessous de la température

Une première modification que j'ai apporté a été de brancher le fil de contraste directement à la masse pour se débarrasser d'un des potentiomètres inutile. Ensuite, j'ai modifier le code à deux relais en ajoutant des espaces à la fin de chaque print du LCD ce qui a corrigé l'un des bugs d'affichage, mais pas le second qui fait que la première valeur de température du capteur DHT est un caractère étrange. J'ai aussi ajouté des actualisation au sein de chaque "while" de la valeur "poten" :

```
#include <Adafruit_Sensor.h>

#include <DHT.h>

#include <LiquidCrystal.h>

#define DHTpin 2          // pin2 devient le pin du DHT11

#define Relai_1 3          // pin3 devient le pin du relai 1

#define Relai_2 4          // pin4 devient le pin du relai 2

#define DHTTYPE DHT22      // on règle le modèle de capteur

DHT dht(DHTpin, DHTTYPE);

LiquidCrystal lcd(7, 8, 9, 10, 11, 12); // connexion des ports de l'écran LCD
```

```

int adcPin = A0;           // attribution du pin analogique A0 comme entrée du signal du potentiomètre
int adcPinfine = A1;       // attribution du pin analogique A1 comme entrée du signal du potentiomètre de
réglage fin
int poten = 0;             // poten est la variable modifiée par les potentiomètres

void setup() {

    pinMode(DHTpin, INPUT); // règle le pin 2 en input

    pinMode(Relai_1, OUTPUT); // règle le pin 3 en output

    pinMode(Relai_2, OUTPUT); // règle le pin 4 en output

    pinMode(adcPin, INPUT); // règle le pin A0 (analogique) comme entrée du signal du potentiomètre

    Serial.begin(9600);     // initialise la comm.

    dht.begin();            // allume le capteur

    lcd.begin(16, 2);

    lcd.setCursor(0, 0);

    lcd.println("Temp =    ");

    lcd.setCursor(0, 1);

    lcd.println("Set =    ");

}

void loop() {

    delay(2000);            //attend un peu entre chaque mesures

    poten = ((7+analogRead(adcPin)/40)+(analogRead(adcPinfine)/125)); // poten prend comme valeur celle
envoyée par le potentiomètre, divisée par 10 pour avoir un interval de température entre 0 et 70°C à laquelle on
ajoute celle d'un deuxième potentiomètre divisé par 100 pour avoir plus de précision

```

```
Serial.println(3+poten);      // on fixe une valeur minimum de T à 3°C de façon à créer un delta en  
additionnant la valeur poten
```

```
lcd.setCursor(7, 0);
```

```
lcd.println(dht.readTemperature());
```

```
lcd.setCursor (12, 0);
```

```
lcd.println(" C ");
```

```
lcd.setCursor (7, 1);
```

```
lcd.println(3+poten);
```

```
lcd.setCursor (9, 1);
```

```
lcd.println (" C  ");
```

```
if ( dht.readTemperature() >= (3+poten) ) {           // si la température est au dessus de (3+poten)
```

```
    while (dht.readTemperature() >= (poten)) {        // et tant qu'elle n'est pas repassée en dessous de  
poten (donc 3°C en dessous de la valeur de déclenchement), on active le ventilateur
```

```
    delay(2000);
```

```
    digitalWrite(Relai_1, LOW);    // le relai se ferme => le ventilateur s'allume
```

```
    Serial.println(dht.readTemperature());
```

```
    Serial.println("closed 1");
```

```
    lcd.setCursor(7, 0);
```

```
    lcd.println(dht.readTemperature());
```

```
    lcd.setCursor (12, 0);
```

```
    lcd.println(" C "); // après chaque print de l'écran LCD j'ai rajouter une instruction pour afficher d'une part  
l'unité mais aussi pour se débarrasser les symboles étranges qui apparaissaient à la fin des print
```

```
poten = ((7+analogRead(adcPin)/40)+(analogRead(adcPinfine)/125)); // Ici j'ai ajouté une actualisation de la
valeur poten afin d'éviter que la valeur ne se fige
```

```
lcd.setCursor (7, 1);
```

```
lcd.println(3+poten);
```

```
lcd.setCursor (9, 1);
```

```
lcd.println (" C  ");
```

```
while (dht.readTemperature() >= (poten+2)) { // et tant qu'elle n'est pas repassée en dessous
de poten (donc 3°C en dessous de la valeur de déclenchement), on active le ventilateur
```

```
delay(2000);
```

```
digitalWrite(Relai_2, LOW); // le relai se ferme => le ventilateur s'allume
```

```
Serial.println(dht.readTemperature());
```

```
Serial.println("closed 2");
```

```
lcd.setCursor(7, 0);
```

```
lcd.println(dht.readTemperature());
```

```
lcd.setCursor (12, 0);
```

```
lcd.println(" C  ");
```

```
poten = ((7+analogRead(adcPin)/40)+(analogRead(adcPinfine)/125));
```

```
lcd.setCursor (7, 1);
```

```
lcd.println(3+poten);
```

```
lcd.setCursor (9, 1);
```

```
lcd.println (" C  ");
```

```

    }

}

else {
    // la température est a 35°C ou moins

    digitalWrite(Relai_1, HIGH);    // le relai s'ouvre => le ventilateur s'arrête

    Serial.println(dht.readTemperature());

    Serial.println("open");

    lcd.setCursor(7, 0);

    lcd.write(dht.readTemperature());

}

}

```

Je n'ai cependant pas réussi à faire fonctionner la communication sérial.

Pour le bug d'affichage de la valeur de température du DHT, il n'est pas toujours présent, notamment, l'affichage se fait correctement si les relais sont éteint (le circuit est fermé). Je suspect donc que le problème viens du fait que les relais sont sur la même alimentation que l'écran. Piste à creuser...

Paul SPIRCKEL :

J'ai continué à améliorer le code. J'ai notamment essayé de modifier les boucles while car lorsque la condition n'est plus valide, les signaux déclarés en LOW ne repassent pas en HIGH. Pour ce faire il faut écrire quelque part "digitalWrite(Relai\_..., HIGH)". Ainsi plutôt que d'imbriquer 2 boucles while, j'ai essayé de faire démarrer les deux ventilateurs tant que la température n'était pas redescendue en dessous de (2+poten) et une fois sorti de la boucle, que le relai 2 s'ouvre mais que le premier reste bien fermé jusqu'à ce que le if ne soit plus vrai. Un deuxième problème s'est alors posé, la sortie d'une boucle while comprise dans un if renvoie immédiatement au else. Même en essayant de mettre un deuxième while dans le if (pour  $T < (2 + \text{poten})$ ), les deux ventilateurs s'éteignent immédiatement.

Ce code n'est clairement pas abouti

```
#include <Adafruit_Sensor.h>

#include <DHT.h>

#include <LiquidCrystal.h>

#define DHTpin 2      // pin2 devient le pin du DHT11

#define Relai_1 4      // pin3 devient le pin du relai 1

#define Relai_2 5      // pin4 devient le pin du relai 2

#define DHTTYPE DHT22    // on règle le modèle de capteur

DHT dht(DHTpin, DHTTYPE);

LiquidCrystal lcd(7, 8, 9, 10, 11, 12); // connexion des ports de l'écran LCD

int adcPin = A0;        // attribution du pin analogique A0 comme entrée du signal du potentiomètre
int adcPinfine = A1;    // attribution du pin analogique A1 comme entrée du signal du potentiomètre de
réglage fin
int poten = 0;          // poten est la variable modifiée par les potentiomètres

void setup() {

  pinMode(DHTpin, INPUT);    // règle le pin 2 en input

  pinMode(Relai_1, OUTPUT);  // règle le pin 3 en output

  pinMode(Relai_2, OUTPUT);  // règle le pin 4 en output

  pinMode(adcPin, INPUT);    // règle le pin A0 (analogique) comme entrée du signal du potentiomètre

  Serial.begin(9600);        // initialise la comm.

  dht.begin();               // allume le capteur

  lcd.begin(16, 2);
```

```

lcd.setCursor(0, 0);

lcd.println("Temp =      ");

lcd.setCursor(0, 1);

lcd.println("Set =      ");


}

void loop() {

    delay(2000);           //attend un peu entre chaque mesures

    poten = ((7+analogRead(adcPin)/40)+(analogRead(adcPinfine)/125)); // poten prend comme valeur celle
    envoyée par le potentiomètre, divisée par 10 pour avoir un interval de température entre 0 et 70°C à laquelle on
    ajoute celle d'un deuxième potentiomètre divisé par 100 pour avoir plus de précision

    Serial.println(3+poten);      // on fixe une valeur minimum de T à 3°C de façon à créer un delta en
    additionnant la valeur poten

    lcd.setCursor(7, 0);

    lcd.println(dht.readTemperature());

    lcd.setCursor (12, 0);

    lcd.println(" C ");

    lcd.setCursor (7, 1);

    lcd.println(3+poten);

    lcd.setCursor (9, 1);

    lcd.println (" C ");

    if ( dht.readTemperature() >= (3+poten) ) {           // si la température est au dessus de (3+poten)

```



```
delay(2000);

Serial.println(dht.readTemperature());

Serial.println("closed 1");

lcd.setCursor(7, 0);

lcd.println(dht.readTemperature());

lcd.setCursor (12, 0);

lcd.println(" C ");

poten = ((7+analogRead(adcPin)/40)+(analogRead(adcPinfine)/125));

lcd.setCursor (7, 1);

lcd.println(3+poten);

lcd.setCursor (9, 1);

lcd.println (" C ");

do {

    delay(2000);

    digitalWrite(Relai_1, LOW);
    digitalWrite(Relai_2, LOW);

    Serial.println(dht.readTemperature());

    Serial.println("closed 1 & 2");

    lcd.setCursor(7, 0);

    lcd.println(dht.readTemperature());
```

```
lcd.setCursor (12, 0);
```

lcd.println(" C "); // après chaque print de l'écran LCD j'ai rajouter une instruction pour afficher d'une part l'unité mais aussi pour se débarrasser des symboles étranges qui apparraissaient à la fin des print

poten = ((7+analogRead(adcPin)/40)+(analogRead(adcPinfine)/125)); // Ici j'ai ajouté une actualisation de la valeure poten afin d'éviter que la valeur ne se fige

```
lcd.setCursor (7, 1);
```

```
lcd.println(3+poten);
```

```
lcd.setCursor (9, 1);
```

```
lcd.println (" C ");
```

```
}
```

while (dht.readTemperature() >= (2+poten)); // et tant qu'elle n'est pas repassée en dessous de poten (donc 3°C en dessous de la valeur de déclenchement), on active le ventilateur

```
do {
```

```
delay(2000);
```

```
digitalWrite(Relai_2, HIGH);
```

```
Serial.println(dht.readTemperature());
```

```
Serial.println("closed 1 & open 2");
```

```
lcd.setCursor(7, 0);
```

```
lcd.println(dht.readTemperature());
```

```
lcd.setCursor (12, 0);
```

lcd.println(" C "); // après chaque print de l'écran LCD j'ai rajouter une instruction pour afficher d'une part l'unité mais aussi pour se débarrasser des symboles étranges qui apparraissaient à la fin des print

poten = ((7+analogRead(adcPin)/40)+(analogRead(adcPinfine)/125)); // Ici j'ai ajouté une actualisation

de la valeur poten afin d'éviter que la valeur ne se fige

```
    lcd.setCursor (7, 1);

    lcd.println(3+poten);

    lcd.setCursor (9, 1);

    lcd.println (" C  ");

}

    while (dht.readTemperature() < (2+poten)); // et tant qu'elle n'est pas repassée en dessous de poten (donc
3°C en dessous de la valeur de déclenchement), on active le ventilateur

}

else {          // la température est à (3+poten) ou moins

    digitalWrite(Relai_1, HIGH);    // le relai 1 s'ouvre => le ventilateur 1 s'arrête
    digitalWrite(Relai_2, HIGH);    // le relai 2 s'ouvre => le ventilateur 2 s'arrête

    Serial.println(dht.readTemperature());

    Serial.println("open");

    lcd.setCursor(7, 0);

    lcd.write(dht.readTemperature());

}

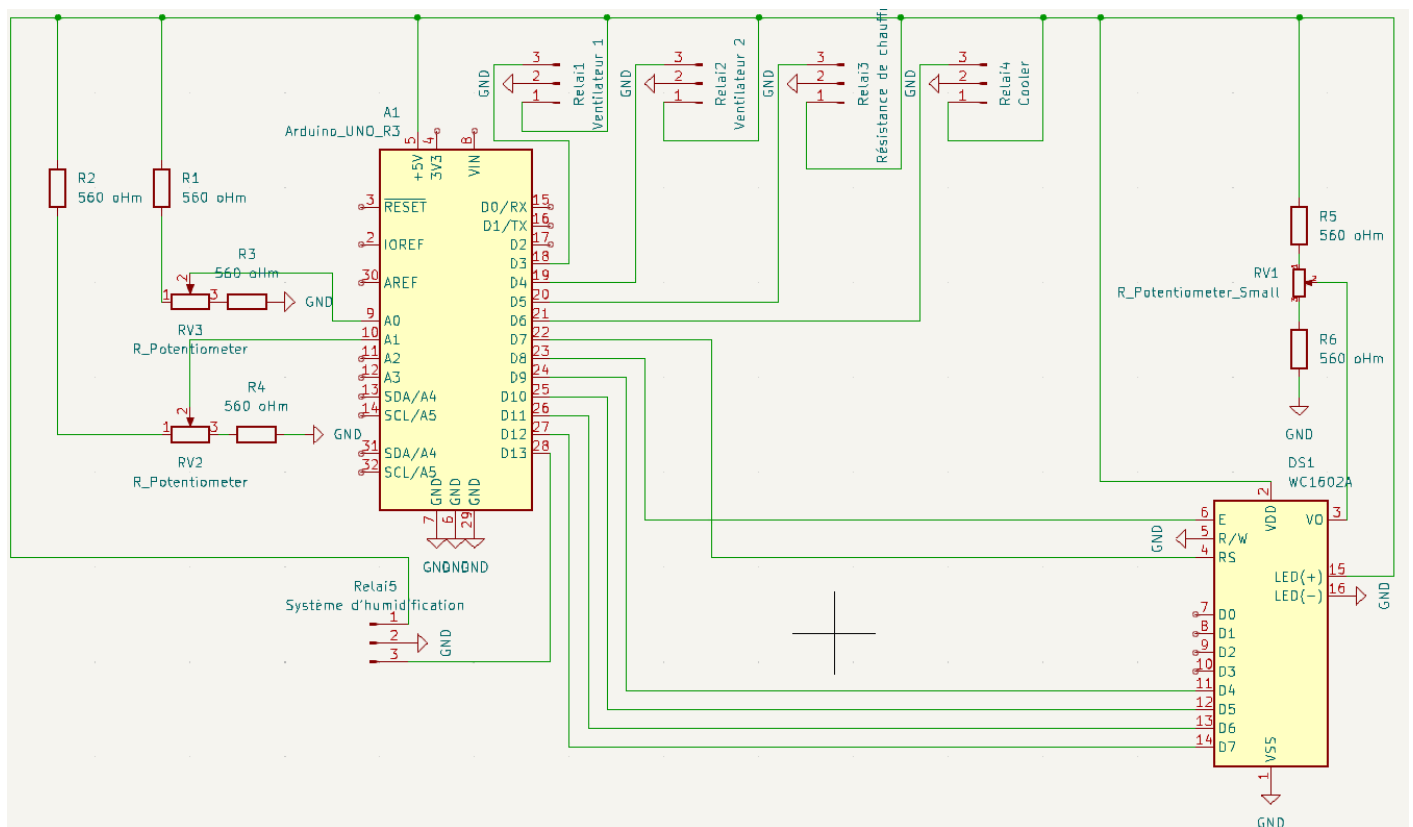
}
```

Miro VON DER BORCH (24/11/2023) :

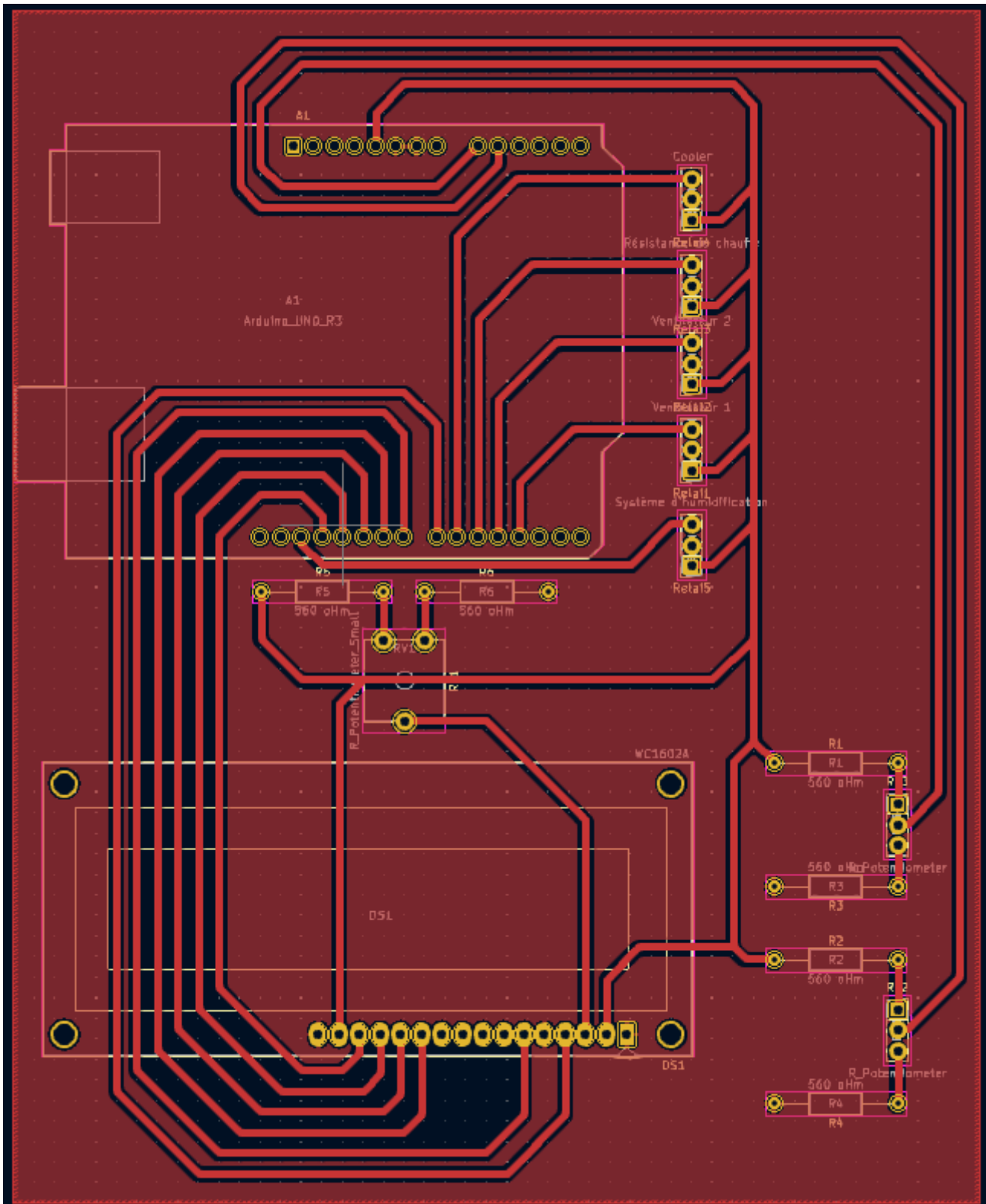
J'ai reproduit le circuit à l'identique dans l'espace prototypage pour pouvoir continuer à avancer dessus. Je l'ai mis dans un des grands bacs en plastique en bas de l'étagère des projets. J'ai ainsi apporté les modifications suivantes au circuit :

- J'ai abandonné le capteur DHT trop peu précis (plusieurs degrés d'incertitude) pour un capteur SHT31 bien plus précis (incertitude de l'ordre du dixième de degré à présent). Cependant, il n'apparaît pas dans le montage ci-dessous car il se branche pour l'instant sur un shield seeeduino via les port I2C. Mais s'il existe un moyen de contourner le shield je suis preneur.
- J'ai échangé les "slide potentiometer" de chez Arduino pour des potentiomètres rotatif plus précis (8 et 10kΩ) et plus esthétique aussi ! Seulement pour cela j'ai dû ajouter quelques résistances pour éviter tout risque de court circuit.
- J'ai aussi ajouté trois relais supplémentaires.
  - Un qui actionnerais une résistance de chauffe afin d'élever la température de la serre au dessus de la température ambiante.
  - Un autre qui actionnerais un système de refroidissement (une pompe à eau par exemple).
  - Un dernier qui actionnerais un système de brumisation afin de pouvoir contrôler l'hygrométrie à terme !
- J'ai aussi remis un tout petit potentiomètre pour la luminosité (c'est finalement plus agréable).

J'ai aussi fait un premier aperçu du circuit imprimé que l'on pourrait faire pour ce montage avec toute les modifications que j'ai mentionné plus haut incluse dedans :



*Le schéma du circuit sur KiCad*



*La PCB*

J'ai joins le fichier de la PCB

Pour ce qui est du code :

- Il faudra maintenant apporter quelques modifications pour l'adapter à la librairie du capteur SHT31 (essentiellement changer les "dht.read" par des "sht.get").

- J'ai connecté
  - "résistance de chauffe" -----> pin digitale 5
  - "cooler" -----> pin digitale 6
  - "système d'humidification" -> pin digital 13
- Il faudrait établir des conditions d'allumage des nouveaux système lié à la régulation de la température, à savoir, le chauffage et le refroidissement actif.
- Il faudrait établir un système de régulation de l'humidité (quand allumer le brumisateur pour humidifier et quand allumer les ventilateurs pour sécher l'air).
- Peut-être aussi que centrer le delta d'allumage autour de la valeur demandé serait plus judicieux pour avoir une meilleur précision. Par ça j'entends, au lieu d'allumer les ventilateurs lorsque la température souhaité est atteinte et les éteindre trois degré en dessous, les allumer un degré au dessus de la valeurs souhaité et les éteindre un degré en dessous de la valeur souhaité

Lors de mes essais pour régler le problème d'affichage qui survenait lorsque les relais s'allumait (Le premier chiffre de la valeur de température lut par le capteur était remplacé par un caractère étrange), je pensai d'abords qu'ils en étaient la cause, mais après quelques essais, les boucles "while", pour une raison que j'ignore, semblent être en cause, en effet le problème survient même si les relais sont déconnectés. Cependant peut-être qu'avec la nouvelle version du code il n'est plus présent.

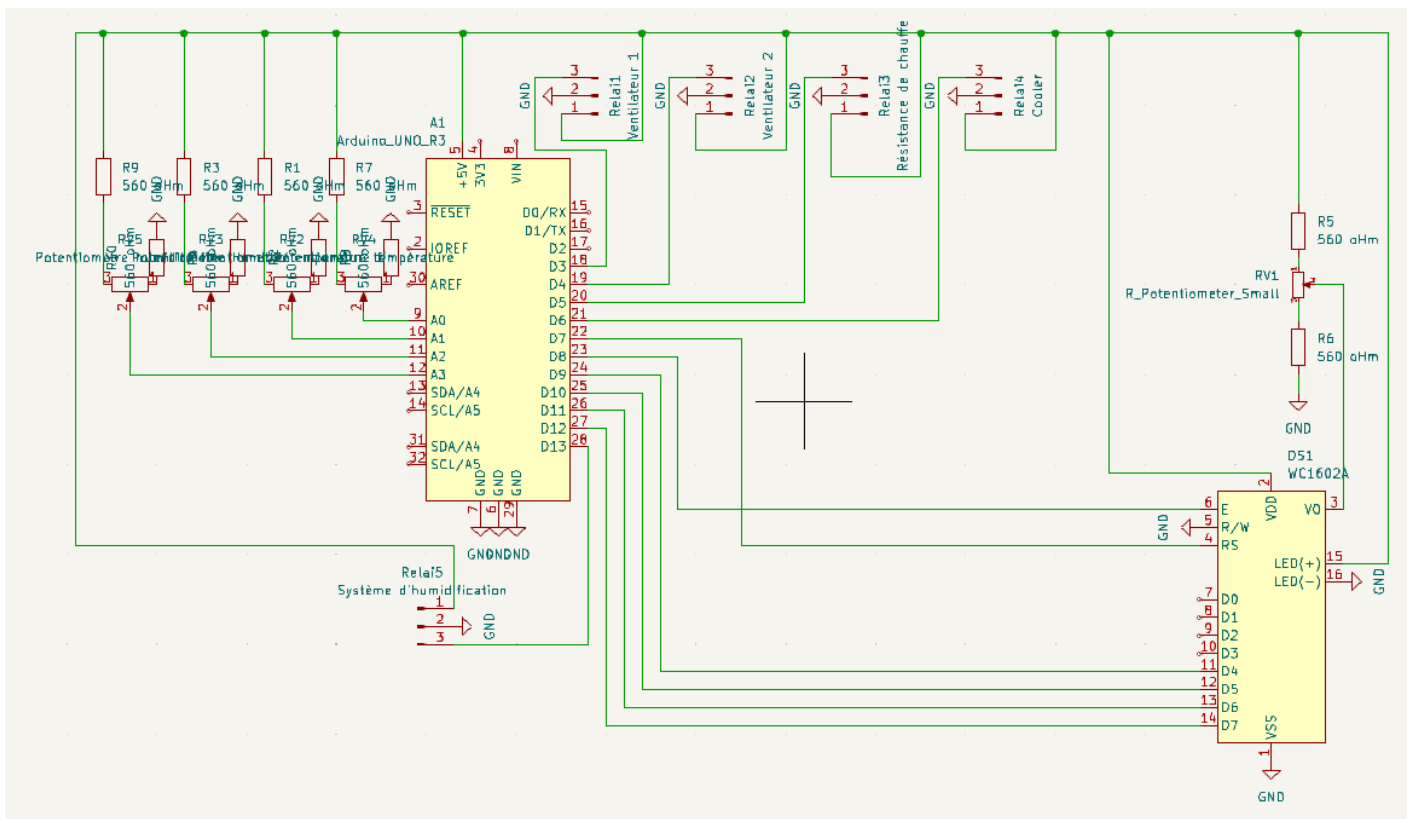
Miro VON DER BORCH (25/11/23) :

J'ai revus un peu la PCB ce matin afin d'ajouter deux potentiomètres en plus sur le circuit actuel, ces potentiomètres devrais servir à contrôler l'humidité souhaité à terme. Je les ai donc branché au pins analogue suivant :

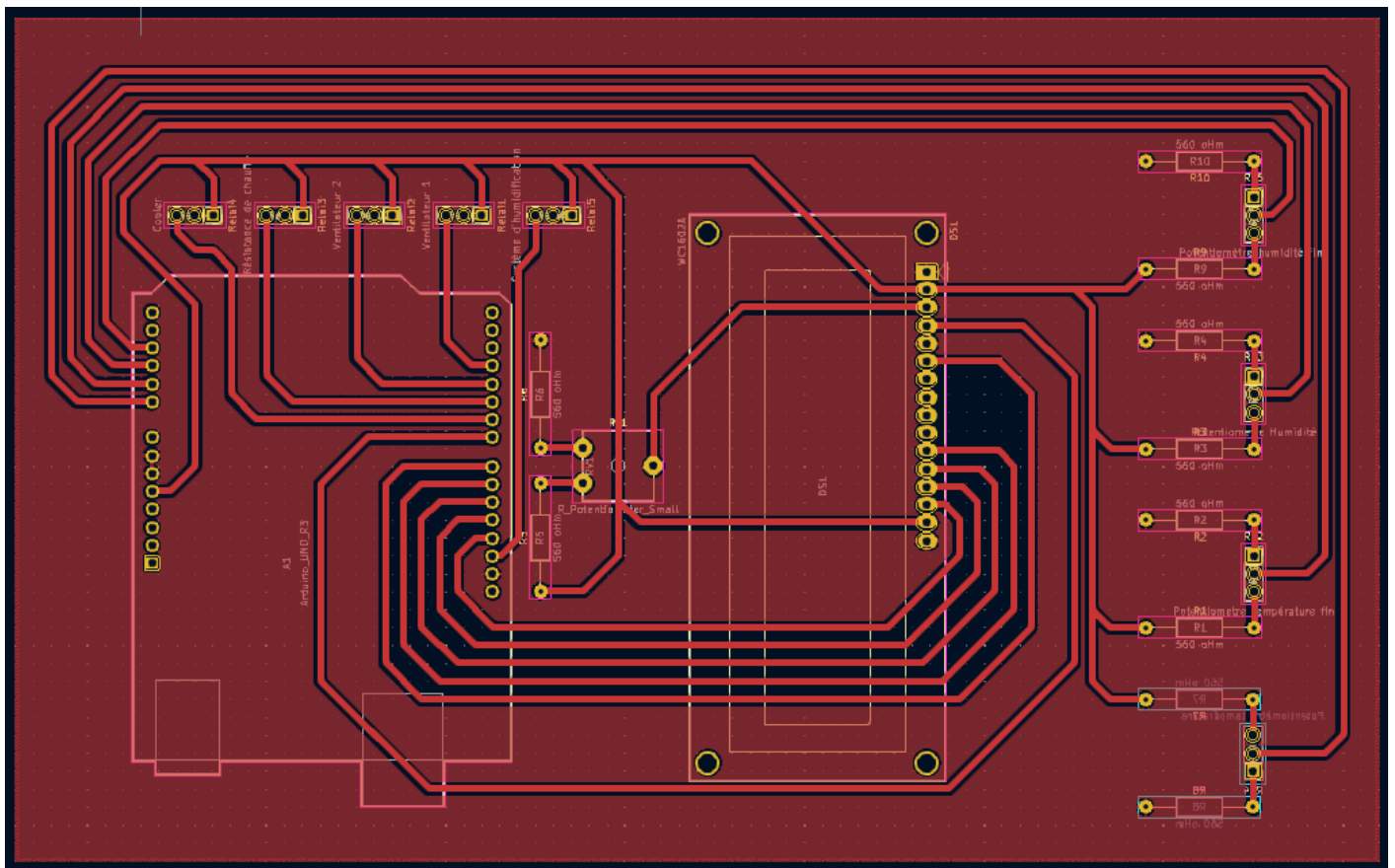
A2 -> Potentiomètre d'humidité

A3 -> Potentiomètre d'humidité fin

Voici les captures d'écran de ce que ça donne et j'ai joins le fichier KiCad :



Le schéma mis à jour



La PCB mis à jour

Paul SPIRCKEL (29/11/2023) :

Après avoir consulté les modifications faites par Miro, je me suis lancé dans l'écriture du programme. Je n'avais pas accès au nouveau circuit donc j'ai programmé à l'aveugle. Afin d'être sûr que toutes les conditions sont lues, j'ai préféré décrire 4 if (trop chaud, trop froid, trop humide, trop sec) avec des deltas (centrés sur la valeur voulue) adaptés à chaque situation (on préférera trop froid plutôt que trop chaud et trop humide plutôt que trop sec). Je propose également d'ajouter au circuit un switch à 3 pins pour choisir entre le refroidissement par ventilation ou par cooler. Je propose les deux codes ci-dessous :

### Ventilation et cooler séparés

```
#include <Adafruit_Sensor.h>

#include <DHT.h>
#include <LiquidCrystal.h>
#include "Arduino.h"
#include <Wire.h>
#include "Adafruit_SHT31.h"
// #include <Adafruit_I2CDevice.h>

#define SHTpin

#define Relai_1 3      // pin3 devient le pin du relai 1 (Ventilateur 1)
#define Relai_2 4      // pin4 devient le pin du relai 2 (Ventilateur 2)
#define Relai_3 5      // pin5 devient le pin du relai 3 (Résistance de chauffe)
#define Relai_4 6      // pin6 devient le pin du relai 4 (Cooler)
#define Relai_5 13     // pin13 devient le pin du relai 5 (Système d'humidification)

LiquidCrystal lcd(7, 8, 9, 10, 11, 12); // connexion des ports de l'écran LCD

int adcTempPin = A0;      // attribution du pin analogique A0 comme entrée du signal du
// potentiomètre de température
int adcTempPinfine = A1;  // attribution du pin analogique A1 comme entrée du signal du
// potentiomètre de réglage fin de température
int Temp_poten = 0;       // Temp_poten est la variable modifiée par les potentiomètres liés à la
// température
int adcHumPin = A2;       // attribution du pin analogique A2 comme entrée du signal du potentiomètre
// d'humidité
int adcHumPinfine = A3;   // attribution du pin analogique A3 comme entrée du signal du
// potentiomètre de réglage fin d'humidité
int Hum_poten = 0;        // Hum_poten est la variable modifiée par les potentiomètres liés à l'humidité
```



```
void setup() {

pinMode(Relai_1, OUTPUT);    // règle le pin 3 en output
pinMode(Relai_2, OUTPUT);    // règle le pin 4 en output
pinMode(Relai_3, OUTPUT);    // règle le pin 5 en output
pinMode(Relai_4, OUTPUT);    // règle le pin 6 en output
pinMode(Relai_5, OUTPUT);    // règle le pin 13 en output


pinMode(adcTempPin, INPUT);   // règle le pin A0 (analogique) comme entrée du signal du potentiomètre
Temp
pinMode(adcTempPinfine, INPUT); // règle le pin A1 (analogique) comme entrée du signal du
potentiomètre Tempfine
pinMode(adcHumPin, INPUT);    // règle le pin A2 (analogique) comme entrée du signal du potentiomètre
Hum
pinMode(adcHumPinfine, INPUT); // règle le pin A3 (analogique) comme entrée du signal du
potentiomètre Humfine


Serial.begin(9600);           // initialise la comm.


//Section consacrée à l'affichage vouée à évoluer en fonction du système d'affichage choisi


lcd.begin(16, 2);

lcd.setCursor(0, 0);

lcd.println("Temp =      ");

lcd.setCursor(0, 1);

lcd.println("Set =      ");

}
```

```

void loop() {

delay(2000);           //attend un peu entre chaque mesures

Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125)); // poten prend
comme valeur celle envoyée par le potentiomètre, divisée par 10 pour avoir un interval de température
entre 0 et 70°C à laquelle on ajoute celle d'un deuxième potentiomètre divisé par 100 pour avoir plus de
précision
Hum_poten = ((7+analogRead(adcHumPin)/40)+(analogRead(adcHumPinfine)/125)); // poten prend
comme valeur celle envoyée par le potentiomètre, divisée par 10 pour avoir un interval de température
entre 0 et 70°C à laquelle on ajoute celle d'un deuxième potentiomètre divisé par 100 pour avoir plus de
précision

serial.print(Temp_poten);

// ---- CONTROLE DE LA TEMPERATURE ----

// Température trop élevée (max 1°C ou dessus de la température souhaitée)

if (sht31.readTemperature() > (1+Temp_poten) ){ // Si la température est supérieure à 1°C au dessus de la
température souhaitée

    while (sht31.readTemperature() >= (Temp_poten-1) { // Le système de refroidissement se met en
marche tant que la température n'est pas redescendue en dessous de 1°C sous Temp_poten

        delay(2000);
        Serial.println(sht31.readTemperature());
        Serial.println("Température trop élevée");
        Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

        digitalWrite(Relai_1, LOW); // Le Relai_1 se ferme (Ventilateur 1 en fonctionnement)
        digitalWrite(Relai_2, LOW); // Le Relai_2 se ferme (Ventilateur 2 en fonctionnement)

    }

}

else {

```

```

    delay(2000);
    Serial.println(sht31.readTemperature());
    Serial.println("Conditions atteintes");
    Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

    digitalWrite(Relai_1, HIGH); // Le Relai_1 s'ouvre (Ventilateur 1 en arrêt)
    digitalWrite(Relai_2, HIGH); // Le Relai_2 s'ouvre (Ventilateur 2 en arrêt)

}

// La température est trop faible (Maximum 3°C en dessous de la température souhaitée, ne pas monter
plus haut que la température souhaitée)

if (sht31.readTemperature() < (Temp_poten-3) ){ // Si la température est inférieure à 3°C en dessous de la
température souhaitée

    while (sht31.readTemperature() < (Temp_poten) { // Le système de chauffage se met en marche tant que
la température n'est pas remontée à Temp_poten

        delay(2000);
        Serial.println(sht31.readTemperature());
        Serial.println("Température trop faible");
        Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

        digitalWrite(Relai_3, LOW); // Le Relai_3 se ferme (Chauffage en fonctionnement)

    }
}

else {

    delay(2000);
    Serial.println(sht31.readTemperature());
    Serial.println("Conditions atteintes");
    Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

    digitalWrite(Relai_3, HIGH); // Le Relai_3 s'ouvre (Chauffage en arrêt)

```

```

}

// Température trop élevée MALGRE LA VENTILATION (plutôt un cas d'été) (max 5°C ou dessus de la
température souhaitée)

if (sht31.readTemperature() > (5+Temp_poten) ){ // Si la température est supérieure à 5°C au dessus de la
température souhaitée

    while (sht31.readTemperature() >= (Temp_poten-1) { // Le système de refroidissement se met en
marche tant que la température n'est pas redescendue en dessous de 1°C sous Temp_poten

        delay(2000);
        Serial.println(sht31.readTemperature());
        Serial.println("Température trop élevée");
        Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

        digitalWrite(Relai_4, LOW); // Le Relai_4 se ferme (Cooler en fonctionnement)

    }

}

else {

    delay(2000);
    Serial.println(sht31.readTemperature());
    Serial.println("Conditions atteintes");
    Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

    digitalWrite(Relai_4, HIGH); // Le Relai_4 s'ouvre (Cooler en arrêt)

}

// ---- CONTROLE DE L'HUMIDITE ----

// Humidité trop élevée

if (sht31.readHumidity() > (Hum_poten+10) ){ // Si l'humidité est 10% au dessus de celle souhaitée

```

```

while (sht31.readHumidity() > (Hum_poten) { // Le système de ventilation se met en marche tant que
l'humidité n'est pas redescuendue jusqu'à Hum_poten

    delay(2000);
    Serial.println(sht31.readHumidity());
    Serial.println("Humidité trop élevée");
    Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

    digitalWrite(Relai_1, LOW); // Le Relai_1 se ferme (Ventilateur 1 en fonctionnement)
    digitalWrite(Relai_2, LOW); // Le Relai_2 se ferme (Ventilateur 2 en fonctionnement)

}
}

else {

    delay(2000);
    Serial.println(sht31.readHumidity());
    Serial.println("Conditions atteintes");
    Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

    digitalWrite(Relai_1, HIGH); // Le Relai_1 s'ouvre (Ventilateur 1 en arrêt)
    digitalWrite(Relai_2, HIGH); // Le Relai_2 s'ouvre (Ventilateur 2 en arrêt)

}

// Humidité trop faible

if (sht31.readHumidity() < (Hum_poten-5) ){ // Si l'humidité est 5% en dessous de celle souhaitée

    while (sht31.readHumidity() <= (Hum_poten) { // Le système d'humidification se met en marche tant que
l'humidité n'est pas remontée à Hum_poten

        delay(2000);
        Serial.println(sht31.readHumidity());
        Serial.println("Humidité trop faible");
        Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

        digitalWrite(Relai_5, LOW); // Le Relai_5 se ferme (Système d'humidification en fonctionnement)

```

```

    }
}

else {

    delay(2000);
    Serial.println(sht31.readHumidity());
    Serial.println("Conditions atteintes");
    Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

    digitalWrite(Relai_5, HIGH); // Le Relai_5 s'ouvre (Système d'humidification à l'arrêt)

}

}

```

## Ventilation ou cooler

```

#include <Adafruit_Sensor.h>

#include <DHT.h>
#include <LiquidCrystal.h>
#include "Arduino.h"
#include <Wire.h>
#include "Adafruit_SHT31.h"
// #include <Adafruit_I2CDevice.h>

#define SHTpin

#define cooling 6          // pin6 devient le pin du qui commande soit la paire de ventilateurs soit le cooler
                           // en fonction de la position de l'interrupteur
#define Relai_1 3         // pin3 devient le pin du relai 1 (Ventilateur 1)
#define Relai_2 4         // pin4 devient le pin du relai 2 (Ventilateur 2)
#define Relai_3 5         // pin5 devient le pin du relai 3 (Résistance de chauffe)
#define Relai_5 13        // pin13 devient le pin du relai 5 (Système d'humidification)

```

```
LiquidCrystal lcd(7, 8, 9, 10, 11, 12); // connexion des ports de l'écran LCD
```

```
int adcTempPin = A0;           // attribution du pin analogique A0 comme entrée du signal du
potentiomètre de température
int adcTempPinfine = A1;       // attribution du pin analogique A1 comme entrée du signal du
potentiomètre de réglage fin de température
int Temp_poten = 0;           // Temp_poten est la variable modifiée par les potentiomètres liés à la
température
int adcHumPin = A2;           // attribution du pin analogique A2 comme entrée du signal du potentiomètre
d'humidité
int adcHumPinfine = A3;       // attribution du pin analogique A3 comme entrée du signal du
potentiomètre de réglage fin d'humidité
int Hum_poten = 0;           // Hum_poten est la variable modifiée par les potentiomètres liés à l'humidité
```

```
void setup() {
```

```
pinMode(cooling, OUTPUT); // règle le pin 6 en output
pinMode(Relai_1, OUTPUT); // règle le pin 3 en output
pinMode(Relai_2, OUTPUT); // règle le pin 4 en output
pinMode(Relai_3, OUTPUT); // règle le pin 5 en output
pinMode(Relai_5, OUTPUT); // règle le pin 13 en output
```

```
pinMode(adcTempPin, INPUT); // règle le pin A0 (analogique) comme entrée du signal du potentiomètre
Temp
pinMode(adcTempPinfine, INPUT); // règle le pin A1 (analogique) comme entrée du signal du
potentiomètre Tempfine
pinMode(adcHumPin, INPUT); // règle le pin A2 (analogique) comme entrée du signal du potentiomètre
Hum
pinMode(adcHumPinfine, INPUT); // règle le pin A3 (analogique) comme entrée du signal du
potentiomètre Humfine
```

```
Serial.begin(9600); // initialise la comm.
```

```
//Section consacrée à l'affichage vouée à évoluer en fonction du système d'affichage choisi
```

```

lcd.begin(16, 2);

lcd.setCursor(0, 0);

lcd.println("Temp =      ");

lcd.setCursor(0, 1);

lcd.println("Set =      ");

}

void loop() {

delay(2000);           //attend un peu entre chaque mesures

Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125)); // poten prend
comme valeur celle envoyée par le potentiomètre, divisée par 10 pour avoir un interval de température
entre 0 et 70°C à laquelle on ajoute celle d'un deuxième potentiomètre divisé par 100 pour avoir plus de
précision
Hum_poten = ((7+analogRead(adcHumPin)/40)+(analogRead(adcHumPinfine)/125)); // poten prend
comme valeur celle envoyée par le potentiomètre, divisée par 10 pour avoir un interval de température
entre 0 et 70°C à laquelle on ajoute celle d'un deuxième potentiomètre divisé par 100 pour avoir plus de
précision

serial.print(Temp_poten);

// ---- CONTROLE DE LA TEMPERATURE ----

// Température trop élevée (max 1°C ou dessus de la température souhaitée)

if (sht31.readTemperature() > (1+Temp_poten) ){ // Si la température est supérieure à 1°C au dessus de la
température souhaitée

    while (sht31.readTemperature() >= (Temp_poten-1) { // Le système de refroidissement se met en
marche tant que la température n'est pas redescendue en dessous de 1°C sous Temp_poten

```



```

    delay(2000);
    Serial.println(sht31.readTemperature());
    Serial.println("Température trop élevée");
    Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

    digitalWrite(cooling, LOW); // Le système de refroidissement sélectionné se met en marche

}

}

else {

    delay(2000);
    Serial.println(sht31.readTemperature());
    Serial.println("Conditions atteintes");
    Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

    digitalWrite(cooling, HIGH); // Le système de refroidissement sélectionné s'arrête

}

// La température est trop faible (Maximum 3°C en dessous de la température souhaitée, ne pas monter
plus haut que la température souhaitée)

if (sht31.readTemperature() < (Temp_poten-3) ){ // Si la température est inférieure à 3°C en dessous de la
température souhaitée

    while (sht31.readTemperature() < (Temp_poten) { // Le système de chauffage se met en marche tant que
la température n'est pas remontée à Temp_poten

        delay(2000);
        Serial.println(sht31.readTemperature());
        Serial.println("Température trop faible");
        Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

        digitalWrite(Relai_3, LOW); // Le Relai_3 se ferme (Chauffage en fonctionnement)

```

```

    }
}

else {

    delay(2000);
    Serial.println(sht31.readTemperature());
    Serial.println("Conditions atteintes");
    Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

    digitalWrite(Relai_3, HIGH); // Le Relai_3 s'ouvre (Chauffage en arrêt)
}

// ---- CONTROLE DE L'HUMIDITE ----

// Humidité trop élevée

if (sht31.readHumidity() > (Hum_poten+10) ){ // Si l'humidité est 10% au dessus de celle souhaitée

    while (sht31.readHumidity() > (Hum_poten) { // Le système de ventilation se met en marche tant que
l'humidité n'est pas redescuendue jusqu'à Hum_poten

        delay(2000);
        Serial.println(sht31.readHumidity());
        Serial.println("Humidité trop élevée");
        Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

        digitalWrite(Relai_1, LOW); // Le Relai_1 se ferme (Ventilateur 1 en fonctionnement)
        digitalWrite(Relai_2, LOW); // Le Relai_2 se ferme (Ventilateur 2 en fonctionnement)

    }
}

else {

    delay(2000);
    Serial.println(sht31.readHumidity());
    Serial.println("Conditions atteintes");

```

```

Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

digitalWrite(Relai_1, HIGH); // Le Relai_1 s'ouvre (Ventilateur 1 en arrêt)
digitalWrite(Relai_2, HIGH); // Le Relai_2 s'ouvre (Ventilateur 2 en arrêt)

}

// Humidité trop faible

if (sht31.readHumidity() < (Hum_poten-5) ){ // Si l'humidité est 5% en dessous de celle souhaitée

    while (sht31.readHumidity() <= (Hum_poten) { // Le système d'humidification se met en marche tant que
l'humidité n'est pas remontée à Hum_poten

        delay(2000);
        Serial.println(sht31.readHumidity());
        Serial.println("Humidité trop faible");
        Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

        digitalWrite(Relai_5, LOW); // Le Relai_5 se ferme (Système d'humidification en fonctionnement)

    }
}

else {

    delay(2000);
    Serial.println(sht31.readHumidity());
    Serial.println("Conditions atteintes");
    Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));

    digitalWrite(Relai_5, HIGH); // Le Relai_5 s'ouvre (Système d'humidification à l'arrêt)

}

}

```

Quelques modifications sont à apporter puisque je n'avais pas le matériel :

- Redéfinir l'échelle des valeurs Temp\_poten correspondant au nouveau potentiomètre
- Créer l'échelle des valeurs (%) Hum\_poten correspondant au nouveau potentiomètre
- Ajouter toutes les lignes d'affichage sur écran
- Essayer d'utiliser des *else if* avec une seule condition *else* qui permet de désactiver tous les relais
- Trouver un moyen de contourner la connexion I2C
- Ajouter toutes les bibliothèques nécessaires si elles n'y sont pas déjà
- Trouver un moyen de déconnecter tous les appareils chauffants si la température est trop élevée pendant trop longtemps
- Ajouter le shield de carte SD et faire en sorte que le programme soit lu dessus

Paul SPIRCKEL (13/12/2023) :

Je n'avais toujours pas le matériel à ma disposition. Je me suis penché sur la possibilité de récupérer le signal I2C du capteur directement en analogique sur la carte Arduino. Ceci ne semble pas possible car une connexion I2C requiert une communication entre 2 composants, typiquement le capteur et le contrôleur Arduino. Il faudra donc probablement ajouter un shield ou autre extension comportant des connexions I2C. Il faudra faire attention à ce que certains pins utilisés pour d'autres fonctions ne gênent pas des connexions du shield, sinon il ne fonctionnera pas. Idem si on utilise le "Wireless SD Shield" pour ajouter de la mémoire. D'ailleurs je me suis renseigné et il semblerait que la carte SD ne puisse être utilisée que pour stocker des données et pas du code. Il doit être possible de faire cela mais avec d'autres composants que je ne connais pas. C'est donc toujours un aspect à développer si un Arduino simple est un peu trop juste (quelques solutions [ici](#) ?). Sinon peut-être serait-il intéressant d'utiliser deux cartes (une pour la température, une pour l'humidité) reliées au même capteur.

J'ai essayé d'écrire le programme avec des else if :

```
#include <Adafruit_Sensor.h>

#include <DHT.h>
#include <LiquidCrystal.h>
#include "Arduino.h"
#include <Wire.h>
#include "Adafruit_SHT31.h"
// #include <Adafruit_I2CDevice.h>

#define SHTpin
```

```
#define Relai_1 3      // pin3 devient le pin du relai 1 (Ventilateur 1)
#define Relai_2 4      // pin4 devient le pin du relai 2 (Ventilateur 2)
#define Relai_3 5      // pin5 devient le pin du relai 3 (Résistance de chauffe)
#define Relai_4 6      // pin6 devient le pin du relai 4 (Cooler)
#define Relai_5 13     // pin13 devient le pin du relai 5 (Système d'humidification)
```

```
LiquidCrystal lcd(7, 8, 9, 10, 11, 12); // connexion des ports de l'écran LCD
```

```
int adcTempPin = A0;      // attribution du pin analogique A0 comme entrée du signal du
potentiomètre de température
int adcTempPinfine = A1;  // attribution du pin analogique A1 comme entrée du signal du
potentiomètre de réglage fin de température
int Temp_poten = 0;       // Temp_poten est la variable modifiée par les potentiomètres liés à la
température
int adcHumPin = A2;       // attribution du pin analogique A2 comme entrée du signal du potentiomètre
d'humidité
int adcHumPinfine = A3;   // attribution du pin analogique A3 comme entrée du signal du
potentiomètre de réglage fin d'humidité
int Hum_poten = 0;        // Hum_poten est la variable modifiée par les potentiomètres liés à l'humidité
```

```
void setup() {
```

```
pinMode(Relai_1, OUTPUT); // règle le pin 3 en output
pinMode(Relai_2, OUTPUT); // règle le pin 4 en output
pinMode(Relai_3, OUTPUT); // règle le pin 5 en output
pinMode(Relai_4, OUTPUT); // règle le pin 6 en output
pinMode(Relai_5, OUTPUT); // règle le pin 13 en output
```

```
pinMode(adcTempPin, INPUT); // règle le pin A0 (analogique) comme entrée du signal du potentiomètre
Temp
pinMode(adcTempPinfine, INPUT); // règle le pin A1 (analogique) comme entrée du signal du
potentiomètre Tempfine
pinMode(adcHumPin, INPUT); // règle le pin A2 (analogique) comme entrée du signal du potentiomètre
Hum
pinMode(adcHumPinfine, INPUT); // règle le pin A3 (analogique) comme entrée du signal du
```

potentiomètre Humfine

```
Serial.begin(9600);      // initialise la comm.
```

```
//Section consacrée à l'affichage vouée à évoluer en fonction du système d'affichage choisi
```

```
lcd.begin(16, 2);
```

```
lcd.setCursor(0, 0);
```

```
lcd.println("Temp =      ");
```

```
lcd.setCursor(0, 1);
```

```
lcd.println("Set =      ");
```

```
}
```

```
void loop() {
```

```
delay(2000);              //attend un peu entre chaque mesures
```

```
Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125)); // poten prend  
comme valeur celle envoyée par le potentiomètre, divisée par 10 pour avoir un interval de température  
entre 0 et 70°C à laquelle on ajoute celle d'un deuxième potentiomètre divisé par 100 pour avoir plus de  
précision
```

```
Hum_poten = ((7+analogRead(adcHumPin)/40)+(analogRead(adcHumPinfine)/125)); // poten prend  
comme valeur celle envoyée par le potentiomètre, divisée par 10 pour avoir un interval de température  
entre 0 et 70°C à laquelle on ajoute celle d'un deuxième potentiomètre divisé par 100 pour avoir plus de  
précision
```

```
serial.print(Temp_poten);
```

```
// ---- CONTROLE DE LA TEMPERATURE ----
```

```
// Température trop élevée (max 1°C ou dessus de la température souhaitée)
```

```
if ((sht31.readTemperature() > (1+Temp_poten)) and (sht31.readTemperature() < (5+Temp_poten))) { // Si la température est supérieure à 1°C au dessus de la température souhaité
```

```
while (sht31.readTemperature() >= (Temp_poten-1) { // Le système de refroidissement se met en marche tant que la température n'est pas redescendue en dessous de 1°C sous Temp_poten
```

```
    delay(2000);
```

```
    Serial.println(sht31.readTemperature());
```

```
    Serial.println("Température trop élevée");
```

```
    Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));
```

```
    digitalWrite(Relai_1, LOW); // Le Relai_1 se ferme (Ventilateur 1 en fonctionnement)
```

```
    digitalWrite(Relai_2, LOW); // Le Relai_2 se ferme (Ventilateur 2 en fonctionnement)
```

```
}
```

```
}
```

```
// La température est trop faible (Maximum 3°C en dessous de la température souhaitée, ne pas monter plus haut que la température souhaitée)
```

```
else if (sht31.readTemperature() < (Temp_poten-3) ){ // Si la température est inférieure à 3°C en dessous de la température souhaitée
```

```
while (sht31.readTemperature() < (Temp_poten) { // Le système de chauffage se met en marche tant que la température n'est pas remontée à Temp_poten
```

```
    delay(2000);
```

```
    Serial.println(sht31.readTemperature());
```

```
    Serial.println("Température trop faible");
```

```
    Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));
```

```
    digitalWrite(Relai_3, LOW); // Le Relai_3 se ferme (Chauffage en fonctionnement)
```

```
}
```

```
}
```

```
// Température trop élevée MALGRE LA VENTILATION (plutôt un cas d'été) (max 5°C ou dessus de la
```

température souhaitée)

```
else if (sht31.readTemperature() > (5+Temp_poten) ){ // Si la température est supérieure à 5°C au dessus  
de la température souhaitée
```

```
    while (sht31.readTemperature() >= (Temp_poten-1) { // Le système de refroidissement se met en  
marche tant que la température n'est pas redescendue en dessous de 1°C sous Temp_poten
```

```
        delay(2000);
```

```
        Serial.println(sht31.readTemperature());
```

```
        Serial.println("Température trop élevée");
```

```
        Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));
```

```
        digitalWrite(Relai_4, LOW); // Le Relai_4 se ferme (Cooler en fonctionnement)
```

```
    }
```

```
}
```

```
// ---- CONTROLE DE L'HUMIDITE ----
```

```
// Humidité trop élevée
```

```
else if (sht31.readHumidity() > (Hum_poten+10) ){ // Si l'humidité est 10% au dessus de celle souhaitée
```

```
    while (sht31.readHumidity() > (Hum_poten) { // Le système de ventilation se met en marche tant que  
l'humidité n'est pas redescuendue jusqu'à Hum_poten
```

```
        delay(2000);
```

```
        Serial.println(sht31.readHumidity());
```

```
        Serial.println("Humidité trop élevée");
```

```
        Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));
```

```
        digitalWrite(Relai_1, LOW); // Le Relai_1 se ferme (Ventilateur 1 en fonctionnement)
```

```
        digitalWrite(Relai_2, LOW); // Le Relai_2 se ferme (Ventilateur 2 en fonctionnement)
```

```
    }
```

```
}
```



```
// Humidité trop faible
```

```
else if (sht31.readHumidity() < (Hum_poten-5) ){ // Si l'humidité est 5% en dessous de celle souhaitée
```

```
    while (sht31.readHumidity() <= (Hum_poten) { // Le système d'humidification se met en marche tant que  
l'humidité n'est pas remontée à Hum_poten
```

```
        delay(2000);
```

```
        Serial.println(sht31.readHumidity());
```

```
        Serial.println("Humidité trop faible");
```

```
        Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));
```

```
        digitalWrite(Relai_5, LOW); // Le Relai_5 se ferme (Système d'humidification en fonctionnement)
```

```
    }
```

```
}
```

```
// Si aucune des conditions précédentes n'est remplie, tout les relais sont ouverts.
```

```
else {
```

```
    delay(2000);
```

```
    Serial.println(sht31.readTemperature());
```

```
    Serial.println(sht31.readHumidity());
```

```
    Serial.println("Conditions atteintes");
```

```
    Hum_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));
```

```
    Temp_poten = ((7+analogRead(adcTempPin)/40)+(analogRead(adcTempPinfine)/125));
```

```
    digitalWrite(Relai_1, HIGH); // Le Relai_1 s'ouvre (Ventilateur 1 à l'arrêt)
```

```
    digitalWrite(Relai_2, HIGH); // Le Relai_2 s'ouvre (Ventilateur 2 à l'arrêt)
```

```
    digitalWrite(Relai_3, HIGH); // Le Relai_3 s'ouvre (Résistance de chauffe à l'arrêt)
```

```
    digitalWrite(Relai_4, HIGH); // Le Relai_4 s'ouvre (Cooler à l'arrêt)
```

```
    digitalWrite(Relai_5, HIGH); // Le Relai_5 s'ouvre (Système d'humidification à l'arrêt)
```

```
}
```

```
}
```

Il reste donc à faire :

- Redéfinir l'échelle des valeurs Temp\_poten correspondant au nouveau potentiomètre
- Créer l'échelle des valeurs (%) Hum\_poten correspondant au nouveau potentiomètre
- Ajouter toutes les lignes d'affichage sur écran
- Corriger le programme (je n'ai pour le moment pas pu me rendre compte de potentielles erreurs)
- Ajouter toutes les bibliothèques nécessaires si elles n'y sont pas déjà
- Trouver un moyen de déconnecter tous les appareils chauffants si la température est trop élevée pendant trop longtemps (fusible ou relai en plus)
- Tester d'empiler l'Arduino Uno + le shield I2C et changer certaines connections pour que rien n'interfère (voir [ici](#) les pins I2C sont ceux de la rangée du bas)
- Utiliser une autre carte ou augmenter la mémoire de programmation si celle de la carte ne suffit pas
- Essayer d'ajouter un switch à 3 pins pour choisir le mode de refroidissement

Paul SPIRCKEL (24/01/2024) :

Je n'avais pas le matériel donc je me suis penché sur l'organisation du boîtier et de la PCB. L'idée est de rendre la PCB très compacte et certainement de désolidariser le LCD de la PCB. J'ai commencé à faire un dessin de découpe de boîte (plexiglas de préférence) qui permet sur une face d'avoir le LCD avec les 4 potentiomètres, sur les tranches, il faudrait mettre des sortes de prises pour connecter les différents appareils que l'on souhaite utiliser (aussi bien pour le côté modulaire de l'appareil que pratique quand il va falloir l'installer). A l'arrière, j'ai prévu un emplacement pour l'alimentation de l'Arduino (à voir si on met un transfo dans le boîtier ou pas, ce qui risque de l'alourdir et de le chauffer). On pourrait rajouter un crochet sur le dessus pour suspendre le boîtier par exemple. Avoir les composants répartis sur différentes faces implique de les désolidariser de la PCB, mais donc de la rendre plus petite (à voir ce qui est le mieux puisque qu'on va devoir ajouter des fils à l'intérieur).

**J'aimerais si possible avoir accès aux dimensions des différents composants pour pouvoir finir ce modèle :)**

Paul SPIRCKEL (31/01/2024) :

J'ai continué le patron du boîtier. J'ai également réorganisé la PCB (fichier en pièce jointe). L'idée est de placer le LCD en haut, les potentiomètres alignés sur le côté, l'Arduino plus ou moins au centre et les relais alignés en bas. De cette manière on aura le plus de composants directement soudés sur la PCB. Il suffira d'adapter le patron du boîtier aux bonnes dimensions. Il faudra aussi intégrer une alimentation 230V pour les relais et une dérivation de courant pour l'Arduino. Je n'ai jamais utilisé KiCad avant donc mon schéma est seulement visuel mais pas du tout fonctionnel. Si

quelqu'un se sent de le faire proprement, ce serait super (je vais essayer d'apprendre d'ici là).

Paul SPIRCKEL (07/02.2024) :

Cette séance je me suis intéressé à la façon dont on pourrait faire varier la tension des panneaux LED. On pourrait utiliser un variateur de tension externe, sauf si l'on veut que la luminosité soit réglée automatiquement (mais est-ce que ça fait du sens ?). J'ai découvert une technique appelée PWM qui permet de générer des signaux analogiques réglables à partir de signaux numériques. Pour autant j'ai l'impression que ça ne génère que des sinusoïdes et pas des tensions constantes, mais la piste reste à creuser. Je n'ai pas pu le tester par manque de temps mais aussi parce que je n'ai jamais travaillé sur un ESP32 et je n'ai pas réussi à corriger des erreurs certainement causées par des problèmes de bibliothèques et autres. J'ai testé le capteur de luminosité, il fonctionne bien.

La boîte du projet est restée en Biologie / Chimie

Miro VON DER BORCH (05/06/2024) :

J'ai beaucoup avancé sur le programme et la PCB du greenduino. Pour l'instant le montage tel quel permet de régler la température, l'humidité, l'arrosage désiré par un système de menu à deux bouton et à l'aide de deux potentiomètres.

J'ai essayé sans grand succès d'ajouter un logbook sur une carte SD et un Backup en cas de coupure de courant, mais pour une raison que j'ignore ça ne fonctionne pas... J'ai donc décidé d'abandonner l'idée du logbook et du backup pour le moment, j'ai laissé en commentaire les bouts de code que j'ai essayé d'ajouter pour faire fonctionner la partie carte SD.

Le code ci dessous est fonctionnel et se comporte correctement, avec le schéma de montage ci joint (les pins digital et analogique ne correspondent pas entre le schéma et le code mais c'est juste que ce sont ceux de la bread board sur laquelle je travaille et que je n'ai pas encore modifié les pins pour les faire correspondre à ceux de la PCB). Le code comporte une phase de menu et une phase d'action. Il prend en entrée les deux boutons, deux potentiomètre d'interface ainsi que le capteur humidité/température haute précision SHT35 et l'interaction se fait via un écran LCD. Il contrôle ainsi indépendamment 6 relais qui sont respectivement : Chauffage, Refroidissement, Ventilation niveau 1, Ventilation niveau 2, Arrosage et Humidification. Les conditions précises de déclenchement peuvent être adapté au besoin dans le code.

Voici le code pour l'instant et des captures d'écrans du schéma et de la PCB :

```
#include <Arduino.h>

#include <Wire.h>

#include <DFRobot_SHT3x.h>
```

```

#include "rgb_lcd.h"

/*#include <SPI.h>

#include <SD.h>*/

#define Fan1 13          // pin3 devient le pin du relai(Fan 1

#define Fan2 12          // pin4 devient le pin du relai(Fan 2

#define Cooler 11          // pin6 devient le pin du Cooler

#define Heating 10          // pin5 devient le pin du chauffage

#define Watering 8          // pin0 devient le pin de l'arrosage

#define Moisturing 9          // pin13 devient le pin de l'humidification

#define OK 5          // pin2 devient le pin du bouton OK

#define Select 6          // pin1 devient le pin du bouton Select

#define Screen 7          // Permet d'éteindre l'écran et économiser de l'énergie

DFRobot_SHT3x  sht3x;          // initie le SHT35

rgb_lcd lcd;

//File GrowBox;

//Sd2Card card;

//SdVolume volume;

//SdFile root;

int GroundSensor = A0;          // attribution du pin analogique A0 comme entrée du signal du capteur d'humidité
du sol

```

```
int Poten = A1;           // attribution du pin analogique A2 comme entrée du signal du potentiomètre
```

```
int PotenFine = A2;       // attribution du pin analogique A3 comme entrée du signal du potentiomètre de  
réglage fin
```

```
// Les variables suivantes sont pour les réglages
```

```
float T = 0;
```

```
float H = 0;
```

```
float W = 0;
```

```
float P = 0;
```

```
float Pf = 0;
```

```
// Les variables suivantes sont pour le backup
```

```
float T1 = 0;
```

```
float H1 = 0;
```

```
float W1 = 0;
```

```
// Les variable suivantes sont pour naviguer entre les menus de réglage
```

```
int Menu = 1;
```

```
int Valider = 0;
```

```
// Les variables suivantes sont pour savoir si les relais sont ouverts ou fermé
```

```
int Fan1Mode = 0;
```

```
int Fan2Mode = 0;
```

```
int CoolerMode = 0;
```

```
int HeatingMode = 0;
```

```
int WateringMode = 0;

int MoisturingMode = 0;

// l'horloge

unsigned long ScreenClock;

//unsigned long LogClock;

unsigned long Clock;

// L'écran

int S = 1;

// Sauvegardes

/*float Log[4];

String dataString = "";

int i = 0;

char C = 0;*/

void setup() {

    sht3x.begin();

    pinMode(Screen, OUTPUT);

    digitalWrite(Screen, HIGH);

    delay (100);
```

```
pinMode(Fan1, OUTPUT);    // règle le pin 3 en output

pinMode(Fan2, OUTPUT);    // règle le pin 4 en output

pinMode(Poten, INPUT);    // règle le pin A0 (analogique) comme entrée du signal du potentiomètre

pinMode(PotenFine, INPUT);

pinMode(Cooler, OUTPUT);

pinMode(Heating, OUTPUT);

pinMode(Watering, OUTPUT);

pinMode(Moisturing, OUTPUT);

pinMode(OK, INPUT_PULLUP);

pinMode(Select, INPUT_PULLUP);

Serial.begin(9600);        // initialise la comm.

while(!Serial){

}

Serial.println();

Serial.println("Serial start");

lcd.begin(16, 2);

Serial.println("init SD card");

lcd.print("Initialization...");

/* if (!card.init(SPI_HALF_SPEED, 4)) {

    Serial.println("init failed");
```

```
while (1);

}*/

Serial.println("init done");


/*if (SD.exists("Log.txt")){

    Serial.println("Log.txt exist");

} else {

    Serial.println("Log.txt doesn't exist");

    GrowBox = SD.open("Log.txt", FILE_WRITE);

    if (GrowBox) {

        GrowBox.println("T;H;W");

        GrowBox.close();

    } else {

        Serial.println("error opening Log.txt");

    }

}

if (SD.exists("Backup.txt")){

    Serial.println("Backup.txt exists");

    GrowBox = SD.open("Backup.txt", FILE_READ);

    GrowBox.seek(0);
```



```
i = 0;

while (GrowBox.available()) {

    C = GrowBox.peek();

    if (C == ';'){

        GrowBox.read();

    } else {

        float newvalue = GrowBox.parseInt();

        Log[i] = newvalue;

        i++;

    }

}

GrowBox.close();

T=Log[0];

H=Log[1];

W=Log[2];

} else {

    Serial.println("Backup.txt not exist");

    GrowBox = SD.open("Backup.txt", FILE_WRITE);

    if (GrowBox) {
```

```
GrowBox.seek(0);
```

```
GrowBox.print("10;0;0");
```

```
GrowBox.close();
```

```
} else {
```

```
    Serial.println("error opening Backup.txt");
```

```
}
```

```
*/
```

```
lcd.clear();
```

```
}
```

```
void loop () {
```

```
    if ((Valider = 0) || (Menu != 0)){
```

```
        Validation();
```

```
    }
```

```
    if (Menu == 1) {
```

```
        lcd.clear();
```

```
        lcd.setCursor(0, 0);
```

```
        lcd.print("Set temp");
```

```
        lcd.setCursor(0, 1);
```

```
lcd.print("Current :");

lcd.setCursor(8, 1);

lcd.print(T);

lcd.setCursor(12, 1);

lcd.print(" C");

delay (500);

lcd.setCursor(15, 1);

lcd.print("X");

while (Menu == 1) {

    if (digitalRead(OK) == LOW) {

        Serial.println("OK");

        Menu = 2;

    }

    if (digitalRead(Select) == LOW) {

        Serial.println("Sel");

        Set_T();

    }

    delay(100);

}

}
```

```
if (Menu == 2) {

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print("Set hum");

    lcd.setCursor(0, 1);

    lcd.print("Current :");

    lcd.setCursor(8, 1);

    lcd.print(H);

    lcd.setCursor(13, 1);

    lcd.print(" %");

    delay (500);

    lcd.setCursor(15, 1);

    lcd.print("X");

    while (Menu == 2) {

        if (digitalRead(OK) == LOW) {

            Serial.println("OK");

            Menu = 3;

        }

    }
```

```
if (digitalRead(Select) == LOW) {  
  
    Serial.println("Sel");  
  
    Set_H();  
  
}  
  
delay(100);  
  
}  
  
}
```

```
if (Menu == 3) {  
  
    lcd.clear();  
  
    lcd.setCursor(0, 0);  
  
    lcd.print("Set watering");  
  
    lcd.setCursor(0, 1);  
  
    lcd.print("Current :");  
  
    lcd.setCursor(8, 1);  
  
    lcd.print(W);  
  
    lcd.setCursor(13, 1);  
  
    lcd.print("%");  
  
    delay (500);  
  
    lcd.setCursor(15, 1);
```

```
lcd.print("X");

while (Menu == 3) {

    if (digitalRead(OK) == LOW) {

        Serial.println("OK");

        Menu = 0;

        Validation();

    }

    if (digitalRead(Select) == LOW) {

        Serial.println("Sel");

        Set_W();
    }

    delay(100);

}

}

if ((Menu == 0) && (Valider == 1)){

    lcd.clear();

    lcd.setCursor(0,0);

    lcd.print("Starting...");
```

```
Serial.print("Starting...");

/*  Log[0] = T;

    Log[1] = H;

    Log[2] = W;

    dataString = "";

    for (int i = 0; i < 2; i++){

        dataString += String(Log[i]);

        if (i < 2){

            dataString += ",";

        }

    }

    GrowBox = SD.open("Backup.csv", O_RDWR);

    if (GrowBox) {

        GrowBox.seek(0);

        GrowBox.print(dataString);

        GrowBox.close();

    } else {

        Serial.println("error opening datalog.txt");

    }

    dataString = "";

    delay(500);*/
```

```
ScreenClock = millis();

// LogClock = millis();

while ((Menu == 0) && (Valider == 1)) {

    Working();

    Clock = millis();

    if ((Clock-ScreenClock >= 10000) && (S == 1)){

        digitalWrite(Screen, 0);

        Serial.println("Screen OFF");

        S = 0;

    }

    if (digitalRead(Select) == LOW) {

        Serial.println("Sel");

        digitalWrite(Screen, HIGH);

        Serial.println("Screen ON");

        S = 1;

        delay(500);

        lcd.begin(16, 2);

        Menu = 1;

    }

}
```



```
if (digitalRead(OK) == LOW) {

    Serial.println("OK");

    digitalWrite(Screen, HIGH);

    Serial.println("Screen ON");

    S = 1;

    delay(500);

    lcd.begin(16, 2);

    ScreenClock = millis();

    Clock = millis();

}

Serial.print("Screen O/I : ");
Serial.println(S);

/*    if (Clock-LogClock >= 900000){

        LogBook();

        LogClock = millis();

    }*/

    delay(500);

}

}

delay (100);

}
```

```
void Ventilation() {

    if ((sht3x.getTemperatureC() >= T) && (Fan1Mode == 0)) {

        digitalWrite(Fan1, LOW);      // le relai se ferme => le ventilateur s'allume

        Fan1Mode = 1;

    }

    if ((sht3x.getTemperatureC() >= T+1) && (Fan1Mode == 1) && (Fan2Mode == 0)) {

        digitalWrite(Fan2, LOW);

        Fan2Mode = 1;

    }

    if ((T >= sht3x.getTemperatureC()) && (Fan2Mode == 1)) {

        digitalWrite(Fan2, HIGH);

        Fan2Mode = 0;

    }

    if (T-1 >= sht3x.getTemperatureC()) {

        digitalWrite(Fan1, HIGH);

        Fan1Mode = 0;

        digitalWrite(Fan2, HIGH);

        Fan2Mode = 0;

    }

}
```

```
}
```

```
Serial.print("Fan1 O/I : ");
```

```
Serial.println(Fan1Mode);
```

```
Serial.print("Fan2 O/I : ");
```

```
Serial.println(Fan2Mode);
```

```
}
```

```
void Chauffage () {
```

```
if ((T-3 >= sht3x.getTemperatureC()) && (HeatingMode == 0)) {
```

```
    digitalWrite(Heating, LOW);
```

```
    HeatingMode = 1;
```

```
}
```

```
if ((sht3x.getTemperatureC() >= T-2) && (HeatingMode == 1)) {
```

```
    digitalWrite(Heating, HIGH);
```

```
    HeatingMode = 0;
```

```
}
```

```
Serial.print("Heating O/I : ");
```

```
Serial.println(HeatingMode);
```

```
}
```

```
void Refroidissement () {
```

```
    if ((sht3x.getTemperatureC() >= T+5) && (CoolerMode == 0)) {
```

```
        digitalWrite(Cooler, LOW);
```

```
        CoolerMode = 1;
```

```
    }
```

```
    if ((T >= sht3x.getTemperatureC()) && (CoolerMode == 1)) {
```

```
        digitalWrite(Cooler, HIGH);
```

```
        CoolerMode = 0;
```

```
    }
```

```
    Serial.print("Cooler O/I : ");
```

```
    Serial.println(CoolerMode);
```

```
}
```

```
void Arrosage () {
```

```
    if (W+10 >= (constrain(map(analogRead(GroundSensor), 670, 440, 0, 100), 0, 100)) && (WateringMode == 0))  
    {
```

```
        digitalWrite(Watering, LOW);
```

```
        WateringMode = 1;
```

```

}

if (((constrain(map(analogRead(GroundSensor), 670, 440, 0, 100), 0, 100)) >= W-10) && (WateringMode ==
1)) {

    digitalWrite(Watering, HIGH);

    WateringMode = 0;

}

Serial.print("Watering O/I : ");
Serial.println(WateringMode);

}

void Humidification () {

if ((H-10 >= sht3x.getHumidityRH()) && (MoisturingMode == 0)) {

    digitalWrite(Moisturing, LOW);

    MoisturingMode = 1;

}

if ((sht3x.getHumidityRH() >= H) && (MoisturingMode == 1)) {

    digitalWrite(Moisturing, HIGH);

    MoisturingMode = 0;

}

Serial.print("Moisturing O/I : ");

```

```
Serial.println(MoisturingMode);

}

void Set_T () {

    Serial.println("Setting Temp");

    T1 = T;

    lcd.clear();

    lcd.setCursor(0,0);

    lcd.print("Wait...");

    delay (500);

    while (Menu == 1) {

        P = map(analogRead(Poten), 60, 996, 100, 500);

        Pf = map(analogRead(PotenFine), 50, 999, -5, 5);

        T = constrain((P+Pf), 100, 500)/10; // poten prend comme valeur celle envoyée par le potentiomètre, divisée
par 10 pour avoir un interval de température entre 0 et 70°C à laquelle on ajoute celle d'un deuxième
potentiomètre divisé par 100 pour avoir plus de précision

        Serial.print("Setting temp to : ");
        Serial.print(T);
        Serial.println("Celsius");

        lcd.clear();

        lcd.setCursor(0, 0);
```

```
lcd.print("Set temp to:");

lcd.setCursor(0, 1);

lcd.print("T=");

lcd.setCursor(2, 1);

lcd.print(T);

lcd.setCursor (7, 1);

lcd.print(" C");

if (digitalRead(OK) == LOW) {

    Serial.println("OK");

    Menu = 2;

}

if (digitalRead(Select) == LOW) {

    Serial.println("Sel");

    Menu = 2;

    T=T1;

}

delay(100);

}

}
```

```
void Set_H () {

    Serial.println("Setting Humidity");

    H1 = H;

    lcd.clear();

    lcd.setCursor(0,0);

    lcd.print("Wait...");

    delay (500);

    while (Menu == 2) {

        P = map(analogRead(Poten), 60, 996, 0, 1000);

        Pf = map(analogRead(PotenFine), 50, 999, -5, 5);

        H = constrain((P+Pf)/10, 0, 100);

        Serial.print("Setting Humidity to : ");
        Serial.print(H);
        Serial.println("%");

        lcd.clear();

        lcd.setCursor(0, 0);

        lcd.print("Set hum to:");

        lcd.setCursor(0, 1);

        lcd.print("H=");

        lcd.setCursor(2, 1);
```



```
lcd.print(H);

lcd.setCursor (8, 1);

lcd.print(" %");

if (digitalRead(OK) == LOW) {

    Serial.println("OK");

    Menu = 3;

}

if (digitalRead(Select) == LOW) {

    Serial.println("Sel");

    Menu = 3;

    H=H1;

}

delay(100);

}

}

void Set_W () {

    Serial.println("Setting Watering");
```

```
W1 = W;

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Wait...");

delay (500);

while (Menu == 3) {

    P = map(analogRead(Poten), 60, 996, 0, 1000);

    Pf = map(analogRead(PotenFine), 50, 999, -5, 5);

    W = constrain((P+Pf)/10, 0, 100);

    Serial.print("Setting Watering to : ");
    Serial.print(W);
    Serial.println("%");

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print("Set watering to:");

    lcd.setCursor(0, 1);

    lcd.print("W=");

    lcd.setCursor(2, 1);

    lcd.print(W);

    lcd.setCursor (7, 1);

    lcd.print("%");
```

```
if (digitalRead(OK) == LOW) {  
  
    Serial.println("OK");  
  
    Menu = 0;  
  
}  
  
if (digitalRead(Select) == LOW) {  
  
    Serial.println("Sel");  
  
    Menu = 0;  
  
    W=W1;  
  
}  
  
    delay(100);  
  
}  
  
}
```

```
void Validation () {  
  
    Valider = 0;  
  
    Menu = 0;  
  
    off();  
  
    Serial.print("Setting : T= ");  
    Serial.print(T);  
    Serial.print("C H= ");  
    Serial.print(H);
```

```
Serial.print("% W=");
Serial.print(W);
Serial.println("%");

lcd.clear();

lcd.setCursor(0, 1);

lcd.print("T=  H=  W= ");

lcd.setCursor(2, 1);

lcd.print(round(T));

lcd.setCursor(7, 1);

lcd.print(round(H));

lcd.setCursor(13, 1);

lcd.print(round(W));

lcd.setCursor(0, 0);

lcd.print("  Valider?  ");

delay (500);

lcd.setCursor(15, 0);

lcd.print("X");

ScreenClock = millis();

while ((Valider == 0) && (Menu == 0)) {

    Clock = millis();

    if ((digitalRead(OK) == LOW) || (Clock-ScreenClock >= 5000)) {
```

```
Serial.println("OK");

Valider = 1;

}

if (digitalRead(Select) == LOW) {

    Serial.println("Sel");

    Menu = 1;

}

delay(100);

}

}
```

```
void off () {

    digitalWrite(Fan1, HIGH);

    digitalWrite(Fan2, HIGH);

    digitalWrite(Heating, HIGH);

    digitalWrite(Cooler, HIGH);

    digitalWrite(Watering, HIGH);

    digitalWrite(Moisturing, HIGH);

    MoisturingMode = 0;
```

```
CoolerMode = 0;

HeatingMode = 0;

Fan1Mode = 0;

Fan2Mode = 0;

WateringMode = 0;

Serial.println("Off");

}

void Working(){

    Serial.print("Setting : T= ");
    Serial.print(T);
    Serial.print("C H= ");
    Serial.print(H);
    Serial.print("% W=");
    Serial.print(W);
    Serial.println("");

    Serial.print("Mesured : T= ");
    Serial.print(sht3x.getTemperatureC());
    Serial.print("C H= ");
    Serial.print(sht3x.getHumidityRH());
    Serial.print("% W=");
    Serial.print(constrain(map(analogRead(GroundSensor), 670, 440, 0, 100), 0, 100));
    Serial.println("");

    lcd.clear();

    lcd.setCursor(0, 1);

    lcd.print("T=  H=  W= ");
```

```
lcd.setCursor(2, 1);

lcd.print(round(T));

lcd.setCursor(7, 1);

lcd.print(round(H));

lcd.setCursor(13, 1);

lcd.print(round(W));

lcd.setCursor(0, 0);

lcd.print("T=  H=  W=  ");

lcd.setCursor(2, 0);

lcd.print(round(sht3x.getTemperatureC()));

lcd.setCursor(7, 0);

lcd.print(round(sht3x.getHumidityRH()));

lcd.setCursor(13, 0);

lcd.print(constrain(map(analogRead(GroundSensor), 670, 440, 0, 100), 0, 100));

Ventilation ();

Chauffage ();

Refroidissement ();

Arrosage ();

Humidification ();

}
```

```
/*void LogBook (){

    Clock = millis();

    T1 = sht3x.getTemperatureC();

    H1 = sht3x.getHumidityRH();

    W1 = constrain(map(analogRead(GroundSensor), 670, 440, 0, 100), 0, 100);

    Log[0] = T;

    Log[1] = H;

    Log[2] = W;

    dataString = "";

    dataString += String(Clock);

    dataString += ",";

    for (int i = 0; i < 2; i++){

        dataString += String(Log[i]);

        if (i < 1){

            dataString += ",";

        }

    }

    GrowBox = SD.open("Backup.csv", FILE_WRITE);

    if (GrowBox) {

        GrowBox.println(dataString);

    }

}
```



```
GrowBox.close();
```

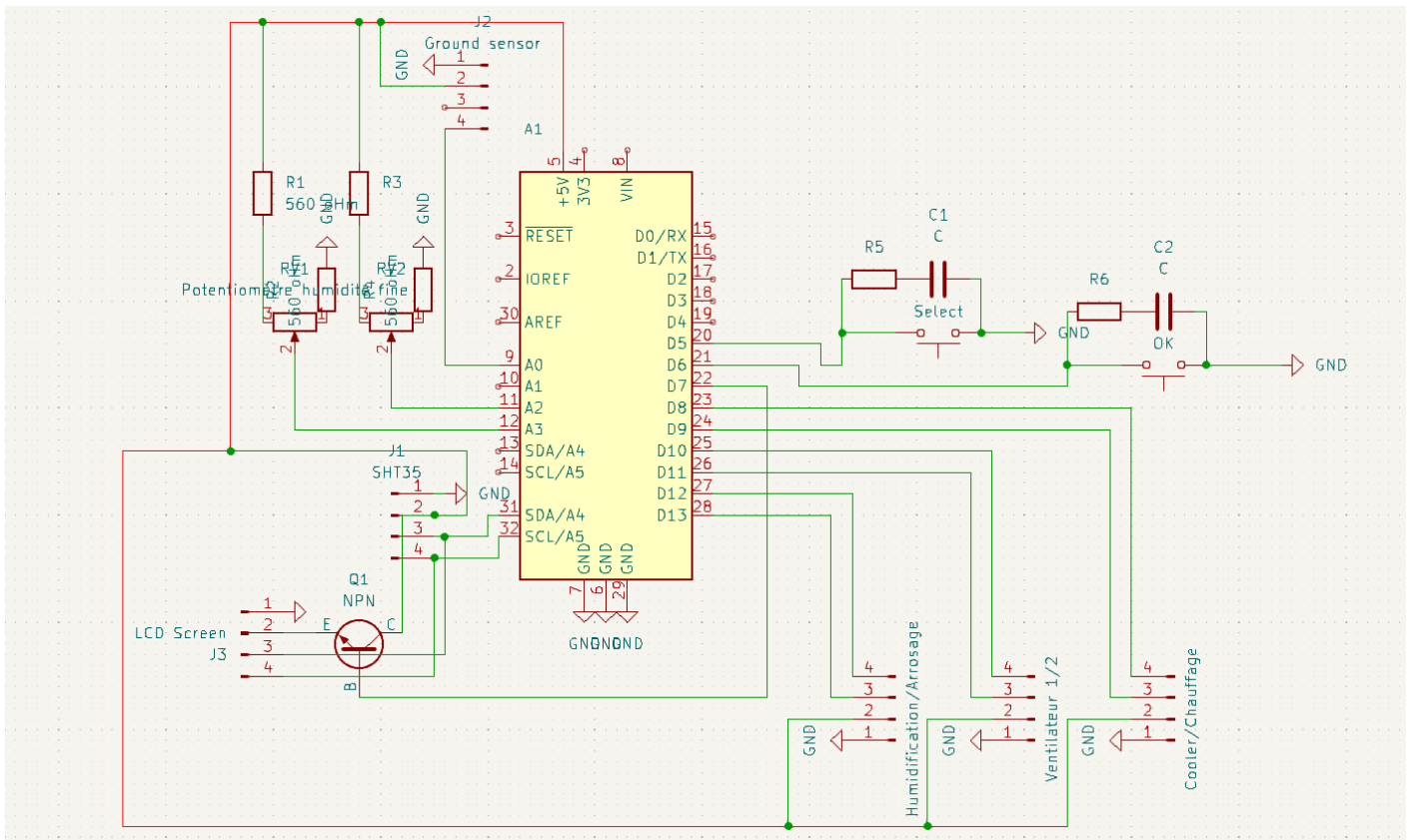
```
} else {
```

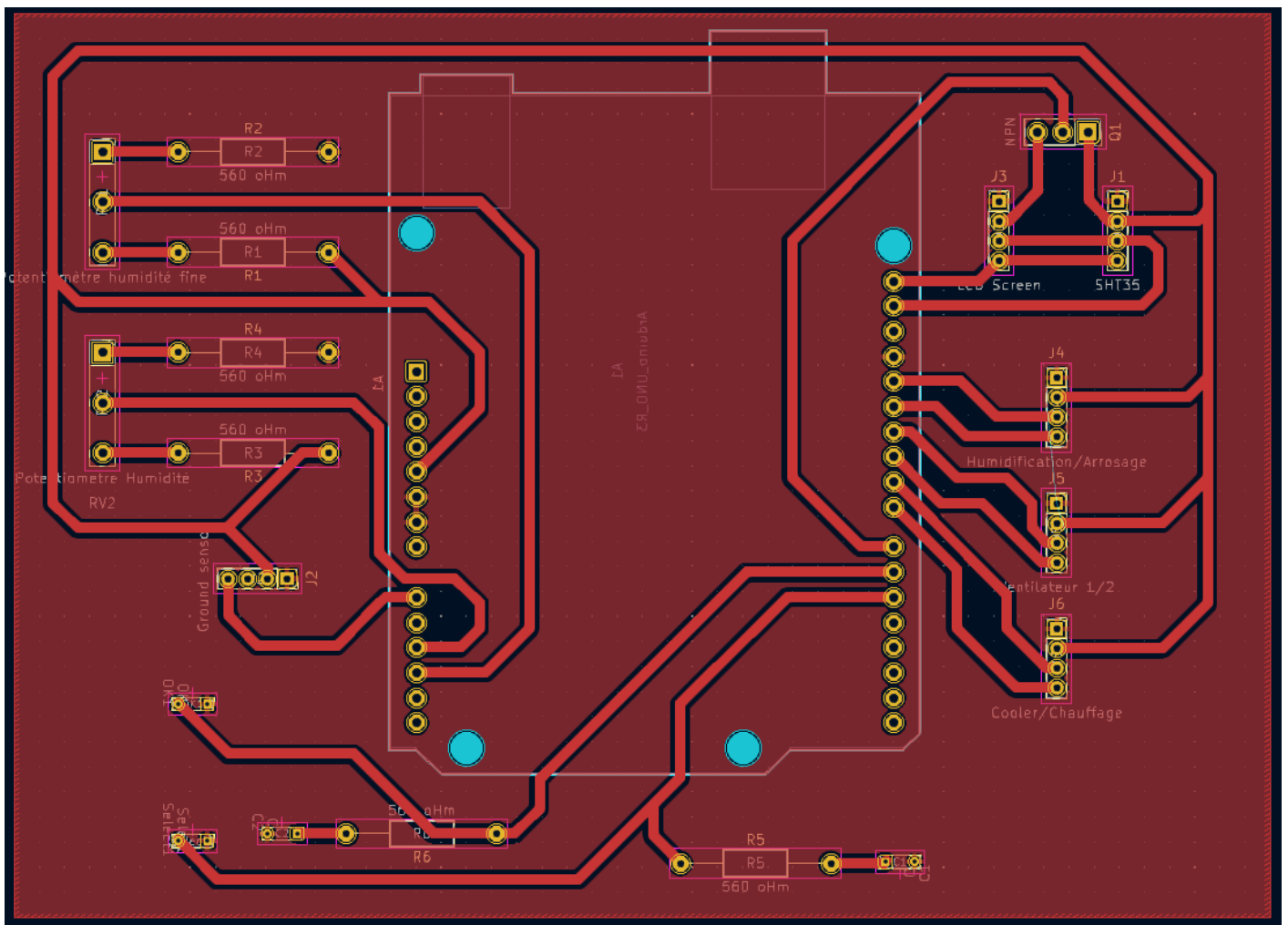
```
Serial.println("error opening datalog.txt");
```

```
}
```

```
dataString = "";
```

```
*/
```



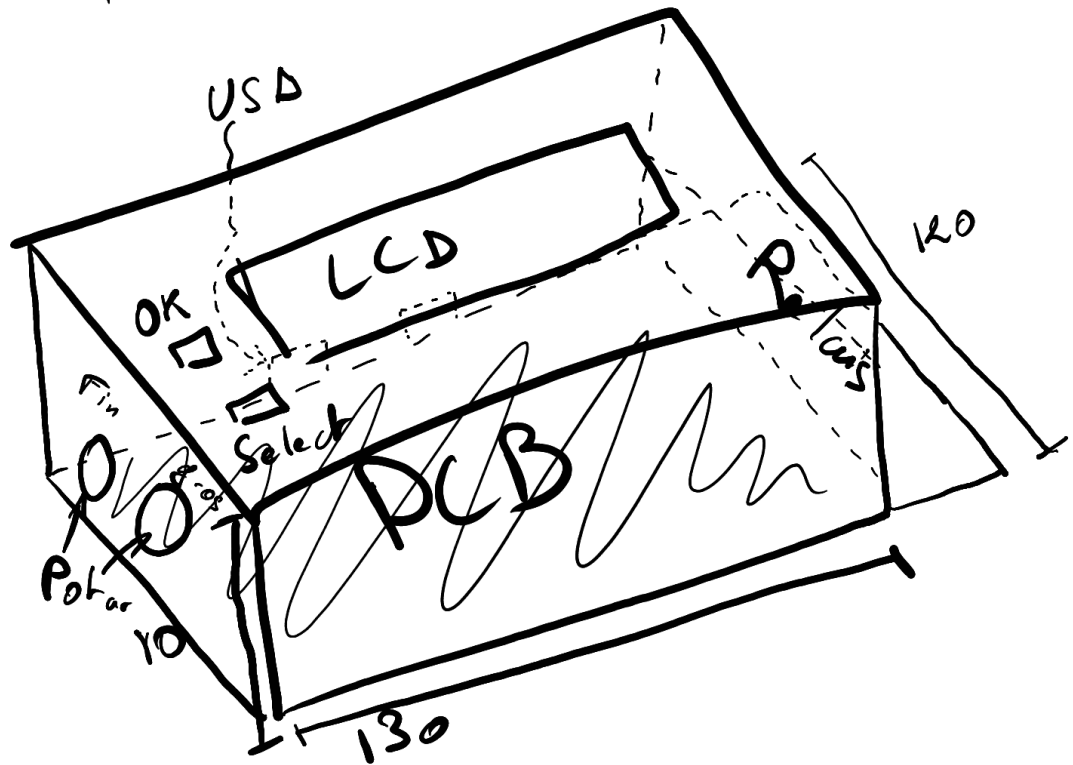


Dimensions : 128 x 91 mm

PS : On pourra noter que la PCB comporte quelques problèmes au niveau du plan de masse qui n'est pas connecté partout (broches "Humidification/arrosage" et "Ventilateur 1/2") mais je pensait régler ce problème après coup avec un bout de fil.

Je travail maintenant sur le design de la boite dans laquelle je veux mettre tout ça. Je pensait à quelques chose de ce type :

boite :



Dimensions : 130(+ la languette avec les relais) x 120 x 100 mm

Alex et Mikhaïl 18/10/2024

Il faudrait simplifier le dispositif afin de le rendre opérationnel le plus rapidement possible, puis itérer avec des versions ultérieures en y ajoutant des fonctionnalités additionnelles.

En premier lieu faire un dispositif avec un écran et des boutons permettant de contrôler la température

Partie du code à faire:

```
bool hysteresisStatusHigh = false
```

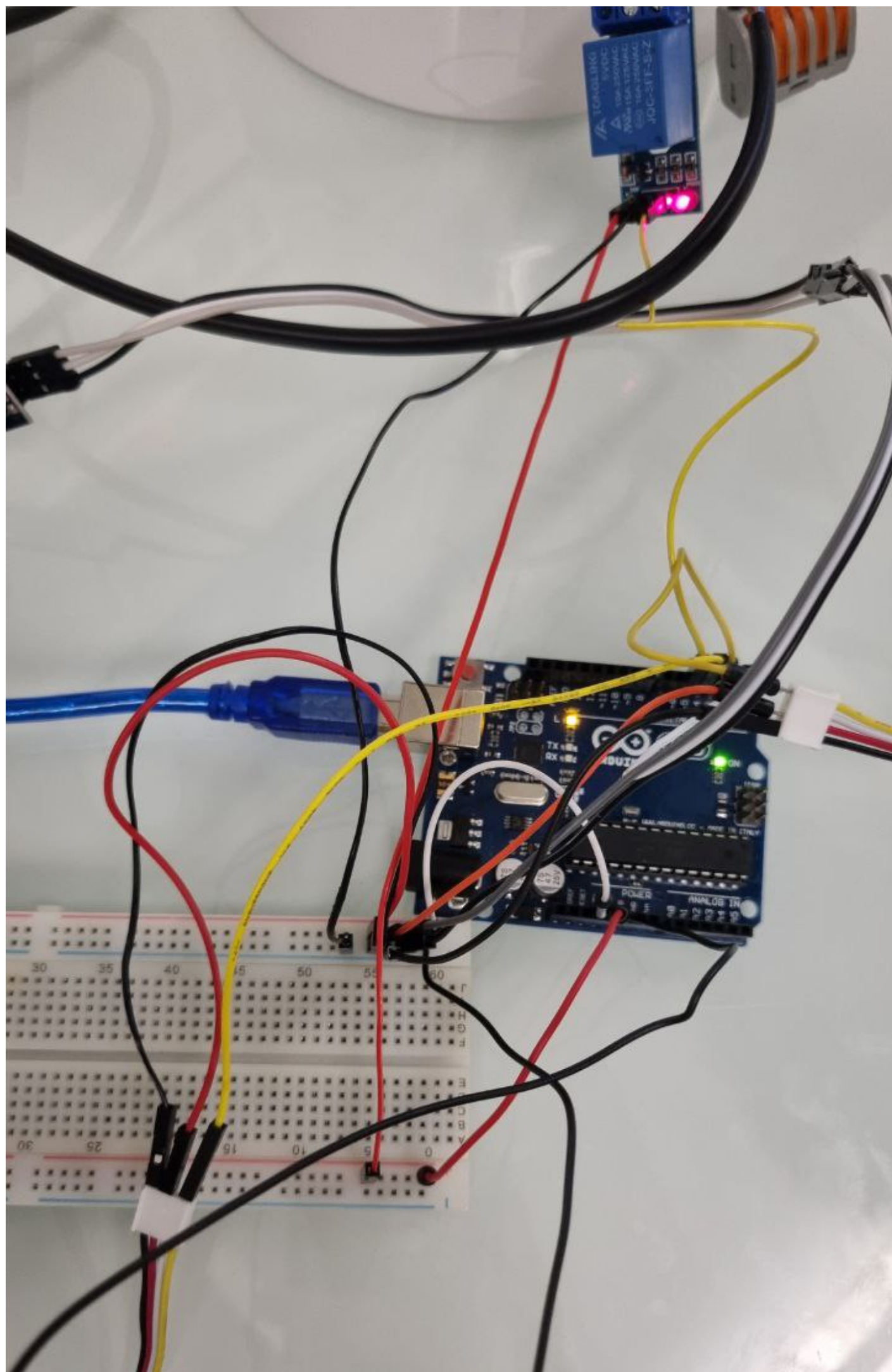
```
bool checkTemperatureSurpassed(){
```

```
    if(hysteresisStatusHigh == false && temperatureGlobal >= TEMPERATUREHIGHPOINT){  
        return(true);
```

```
}  
if(hysteresisStatusHigh && temperatureGlobal >= TEMPERATURELOWPOINT){  
    return(true);  
}  
return(false);  
}  
  
void getTemperature(){  
    temperatureStatus = checkTemperatureSurpassed();  
}
```

25/10/2024 Mikhaïl et Irina

Le code pour activer le ventilateur (branché sur le pin 5) en fonction de la température ambiante (thermomètre branché sur le pin 2 ), contrôlé par les boutons (branchés sur des pins 3 et 4)



```
#include "DHT.h"

#define DHTPIN 2
#define DHTTYPE DHT21

DHT dht(DHTPIN, DHTTYPE);

#define BUTTON_INCREASE 3
#define BUTTON_DECREASE 4
#define FUN_PIN 5

float targetTemperature = 25.0;
float temperature;

void setup() {
  Serial.begin(9600);
  dht.begin();

  pinMode(BUTTON_INCREASE, INPUT_PULLUP);
  pinMode(BUTTON_DECREASE, INPUT_PULLUP);
  pinMode(FUN_PIN, OUTPUT);
}

void loop() {
  float temperature = dht.readTemperature();

  Serial.print("Current temperature:");
  Serial.print(temperature);
  delay(1000);

  if (digitalRead(BUTTON_INCREASE) == LOW){
    targetTemperature += 0.5;
    Serial.print("Target temperature:");
    Serial.println(targetTemperature);
  }

  if (digitalRead(BUTTON_DECREASE) == LOW){
    targetTemperature -= 0.5;
    Serial.print("Target temperature:");
```

```
Serial.println(targetTemperature);  
}  
  
if (temperature > targetTemperature){  
  digitalWrite(FUN_PIN,LOW);  
} else if(temperature < targetTemperature - 1.0){  
  digitalWrite(FUN_PIN,HIGH);  
}  
  
}
```

04/12/2024 Alex et Iryna

Nouveau wiki ---> <https://wiki.fablab.sorbonne-universite.fr/BookStack/books/systeme-ventilation-fablab>

---

Revision #7

Created 19 May 2024 11:22:56 by Ouerfili Chaima

Updated 4 December 2024 16:49:25 by Iryna