```
#include <LiquidCrystal.h>
#include <Wire.h>
#include <PinChangeInt.h>
#include "DHT.h"
#define DHTPIN 13 // capteur DHT22
#define DHTTYPE DHT22 // DHT 22 (AM2302)
DHT dht(DHTPIN, DHTTYPE); //déclaration du capteur

//Branchements
const int Sortie1 = 6; //Relais 1
const int Sortie2 = 7; //Relais 2
const int Sortie3 = 8; //Relais 3
const int Sortie4 = 9; //Relais 4
const int BP_Validation = A2 ; //Bouton de validation
const int codA = A3; //Encodeur
const int codB = A4; //Encodeur

//Brochage LCD
LiquidCrystal lcd(11,10,2,3,4,5); //Liaison 4 bits de données

char ligne1[21] = ""; //Ligne 1 du lcd
char ligne2[21] = ""; //ligne 2 du lcd
char ligne3[21] = ""; //Ligne 3 du lcd
char ligne4[21] = ""; //ligne 4 du lcd

//Capteurs
int Text; //Température extérieure
int Tint; //Température intérieure
int Hint; //Hygrométrie intérieure
int Tmin = 99, Tmax = 0, Hmin = 99, Hmax = 0; //Extrêmes
int T1min, T2min, T3min, T4min; //Consignes basses température
int T1max = 99, T2max = 99, T3max = 99, T4max = 99; //Consignes hautes température
int H1min, H2min, H3min, H4min; //Consignes basses humidité
int H1max = 99, H2max = 99, H3max = 99, H4max = 99; //Consignes hautes humidité

//Encodeur
int Menu = 1; //Affiche le premier menu
int Valid = 0; //Affiche la suite du premier menu
int valCod = 0; //Position initiale de l'encodeur
// variables pour les routines d'interruption
boolean A_set = false;
boolean B_set = false;
boolean A_change = false;
boolean B_change = false;
static boolean rotation = false;// gestion de l'anti-rebonds
int etatCodA = LOW; //Etat de l'encodeur A
int etatCodB = LOW; //Etat de l'encodeur B
int etatBP_Validation = LOW; //Etat du bouton de validation

//Initialisation des commandes relais
int etatRelais1 = LOW;
int etatRelais2 = LOW;
int etatRelais3 = LOW;
int etatRelais4 = LOW;

// Temporisation
int h_onTime1, m_onTime1, h_offTime1, m_offTime1;
int h_onTime2, m_onTime2, h_offTime2, m_offTime2;
int h_onTime3, m_onTime3, h_offTime3, m_offTime3;
int h_onTime4, m_onTime4, h_offTime4, m_offTime4;
unsigned long onTime1, offTime1;
unsigned long onTime2, offTime2;
unsigned long onTime3, offTime3;
unsigned long onTime4, offTime4;
```

```
unsigned long previousMillis1 = 0;
unsigned long previousMillis2 = 0;
unsigned long previousMillis3 = 0;
unsigned long previousMillis4 = 0;

//--------------------------------------------------------------------------------
void setup()
{
  Serial.begin(9600); // Initialisation la communication avec le PC
  analogReference(INTERNAL); // Amélioration de la précision en réduisant la plage de
mesure

  //paramétrage des entrées/sorties
  pinMode(Sortie1, OUTPUT);
  pinMode(Sortie2, OUTPUT);
  pinMode(Sortie3, OUTPUT);
  pinMode(Sortie4, OUTPUT);
  pinMode(BP_Validation, INPUT);
  pinMode(codA, INPUT);
  pinMode(codB, INPUT);
  //digitalWrite(codA, HIGH); //Active la pull-up interne de l'arduino
  //digitalWrite(codB, HIGH); //Active la pull-up interne de l'arduino
  PCintPort::attachInterrupt(codA, EncodeurA, CHANGE); //interruption à chaque changement
de front sur l'encodeur A
  PCintPort::attachInterrupt(codB, EncodeurB, CHANGE); //interruption à chaque changement
de front sur l'encodeur B

  //paramétrage du LCD et intro
  lcd.begin(20,4); // Ecran 20 colonnes et 4 lignes
  lcd.setCursor(6,1);
  lcd.print("Bonjour!"); // Message de bienvenue
  lcd.setCursor(3,3);
  lcd.print("Version 0.5");
  delay(2000); // Attente de 2s
  lcd.clear(); // Effacement de l'écran
}
//--------------------------------------------------------------------------------
void loop()
{
  Humidite(); //Mesure de l'hygrométrie
  Temperature(); //Mesure de la température
  Affichage(); //Gestion de l'affichage des menus sur le LCD
  Validation(); //Action du bouton de validation
  EncodeurA(); //Action de l'encodeur A incrémentation
  EncodeurB(); //Action de l'encodeur B décrémentation
  Regulation(); //Régulation en température et hygrométrie
}
//--------------------------------------------------------------------------------
int Temperature ()
{
  //Lecture et conversion de la tension en température
  int analogTemp = analogRead(A0);
  Text = analogTemp * (1.1 / 1023.0 * 100.0);
}
//--------------------------------------------------------------------------------
int Humidite ()
{
  //Lecture de l'hygrométrie et de la température (en celsius par defaut)
  Hint = dht.readHumidity();
  Tint = dht.readTemperature();

  //Gestion des extrêmes
  if (Tint < Tmin) {Tmin = Tint;} //température minimale
  if (Tint > Tmax) {Tmax = Tint;} //température maximale
```

```
  if (Hint < Hmin) {Hmin = Hint;} //hygrométrie minimale
  if (Hint > Hmax) {Hmax = Hint;} //hygrométrie maximale

  if (isnan(Hint) || isnan(Tint)) //Vérification de la lecture des données
  {
    Serial.println("Erreur de lecture DHT");//message si lecture erronée
  }
}
//-------------------------------------------------------------------------------
void EncodeurA()
{
  etatCodA = digitalRead(codA); //Lecture de la broche A de l'encodeur
  if (rotation) delay(1); // Anti-rebond
  rotation= true; //declenchement de l'anti-rebond

  if (etatCodA != A_set) //Detection d'une rotation
  {
    A_set = !A_set;
    if (B_change)
    {
      if (Valid == 0)
      {
        Menu++;
      }
      else
      {
        valCod++;
      }
      B_change = false;
    }
    else
    {
      A_change = true;
    }
  }
  rotation = false;
  if(Menu >= 5) // on revient au premier menu après le dernier
  {
    Menu = 0;
    Valid = 0;
  }
}
//-------------------------------------------------------------------------------
void EncodeurB()
{
  etatCodB = digitalRead(codB); //Lecture de la broche B de l'encodeur
  if (rotation) delay(5); // Anti-rebond
  rotation= true; //declenchement de l'anti-rebond

  if (etatCodB != B_set) //Detection d'une rotation
  {
    B_set = !B_set;
    if (A_change)
    {
      if (Valid == 0)
      {
        Menu--;
        delay(100);
      }
      else
      {
        valCod--;
        delay(100);
      }
```

```
      }
      else
      {
        B_change = true;
      }
    }
    rotation = false;
    if(Menu <= -1) // on revient au dernier menu avant le premier
    {
      Menu = 4;
      Valid = 0;
    }
}
//--------------------------------------------------------------------------------
int Validation ()
{
  etatBP_Validation = digitalRead(BP_Validation); //Lecture de l'état du bouton de
validation

  //Permet de compter le nombre de pression sur le bouton de validation
  if(etatBP_Validation == LOW && Valid < 9)
  {
    Valid++;
    delay(100);
  }
  else if(Valid == 9) //RAZ une fois les consignes remplies
  {
    Menu = 0;
    Valid = 0;
  }
}
//--------------------------------------------------------------------------------
int Affichage()
{
  delay(100); //Evite le scintillement de l'écran (du moins on ne le voie pas..)

  //Affichage du menu correspondant au bon nombre de pression sur le bouton de sélection
  switch(Menu)
  {
    case 0 :
    if(Valid == 1)
    {
      sprintf(ligne1, "Tmin:%2dC    Hmin:%2d%%", Tmin, Hmin);
      sprintf(ligne2, "Tmax:%2dC    Hmax:%2d%%", Tmax, Hmax);
      sprintf(ligne3, "    -- option 1--");
      sprintf(ligne4, "    -- option 2--");
      break;
    }
    else if(Valid == 0 || Valid == 2)
    {
      sprintf(ligne1, "Humi int:%d%%",Hint);
      sprintf(ligne2, "Temp int:%dC",Tint);
      sprintf(ligne3, "");
      sprintf(ligne4, "Temp ext:%dC", Text);
      Valid = 0;
      valCod = 0;
      break;
    }

    case 1 :
      sprintf(ligne1, "----- Relais %d -----", Menu);
      sprintf(ligne2, "     %2d<Temp<%2d", T1min, T1max);
      sprintf(ligne3, "     %2d<Humi<%2d", H1min, H1max);
```

```
        sprintf(ligne4, "%2dh%2dmin / %2dh%2dmin",h_onTime1, m_onTime1, h_offTime1,
m_offTime1);
      T1min = constrain(T1min, 0, T1max - 1);
      T1max = constrain(T1max, T1min + 1, 99);
      H1min = constrain(H1min, 0, H1max - 1);
      H1max = constrain(H1max, H1min + 1, 99);
      h_onTime1 = constrain(h_onTime1, 0, 23);
      m_onTime1 = constrain(m_onTime1, 0, 59);
      h_offTime1 = constrain(h_offTime1, 0, 23);
      m_offTime1 = constrain(m_offTime1, 0, 59);
      break;

    case 2 :
      sprintf(ligne1, "----- Relais %d -----", Menu);
      sprintf(ligne2, "     %2d<Temp<%2d", T2min, T2max);
      sprintf(ligne3, "     %2d<Humi<%2d", H2min, H2max);
      sprintf(ligne4, "%2dh%2dmin / %2dh%2dmin",h_onTime2, m_onTime2, h_offTime2,
m_offTime2);
      T2min = constrain(T2min, 0, T2max - 1);
      T2max = constrain(T2max, T2min + 1, 99);
      H2min = constrain(H2min, 0, H2max - 1);
      H2max = constrain(H2max, H2min + 1, 99);
      h_onTime2 = constrain(h_onTime2, 0, 23);
      m_onTime2 = constrain(m_onTime2, 0, 59);
      h_offTime2 = constrain(h_offTime2, 0, 23);
      m_offTime2 = constrain(m_offTime2, 0, 59);
      break;

    case 3 :
      sprintf(ligne1, "----- Relais %d -----", Menu);
      sprintf(ligne2, "     %2d<Temp<%2d", T3min, T3max);
      sprintf(ligne3, "     %2d<Humi<%2d", H3min, H3max);
      sprintf(ligne4, "%2dh%2dmin / %2dh%2dmin",h_onTime3, m_onTime3, h_offTime3,
m_offTime3);
      T3min = constrain(T3min, 0, T3max - 1);
      T3max = constrain(T3max, T3min + 1, 99);
      H3min = constrain(H3min, 0, H3max - 1);
      H3max = constrain(H3max, H3min + 1, 99);
      h_onTime3 = constrain(h_onTime3, 0, 23);
      m_onTime3 = constrain(m_onTime3, 0, 59);
      h_offTime3 = constrain(h_offTime3, 0, 23);
      m_offTime3 = constrain(m_offTime3, 0, 59);
      break;

    case 4 :
      sprintf(ligne1, "----- Relais %d -----", Menu);
      sprintf(ligne2, "     %2d<Temp<%2d", T4min, T4max);
      sprintf(ligne3, "     %2d<Humi<%2d", H4min, H4max);
      sprintf(ligne4, "%2dh%2dmin / %2dh%2dmin",h_onTime4, m_onTime4, h_offTime4,
m_offTime4);
      T4min = constrain(T4min, 0, T4max - 1);
      T4max = constrain(T4max, T4min + 1, 99);
      H4min = constrain(H4min, 0, H4max - 1);
      H4max = constrain(H4max, H4min + 1, 99);
      h_onTime4 = constrain(h_onTime4, 0, 23);
      m_onTime4 = constrain(m_onTime4, 0, 59);
      h_offTime4 = constrain(h_offTime4, 0, 23);
      m_offTime4 = constrain(m_offTime4, 0, 59);
      break;
  }
  if(Menu != 0)
  {
    switch(Valid)
    {
```

```
      case 1 :
        lcd.setCursor(6,1);
        lcd.blink();
        delay(100);
        if (EncodeurA || EncodeurB)
        {
          switch(Menu)
          {
            case 1 :
            T1min = T1min + valCod;
            break;

            case 2 :
            T2min = T2min + valCod;
            break;

            case 3 :
            T3min = T3min + valCod;
            break;

            case 4 :
            T4min = T4min + valCod;
            break;
          }
        }
        valCod = 0;
        break;

      case 2 :
        lcd.setCursor(14,1);
        lcd.blink();
        delay(100);
        if (EncodeurA || EncodeurB)
        {
          switch(Menu)
          {
            case 1 :
            T1max = T1max + valCod;
            break;

            case 2 :
            T2max = T2max + valCod;
            break;

            case 3 :
            T3max = T3max + valCod;
            break;

            case 4 :
            T4max = T4max + valCod;
            break;
          }
        }
        valCod = 0;
        break;

      case 3 :
        lcd.setCursor(6,2);
        lcd.blink();
        delay(100);
        if (EncodeurA || EncodeurB)
        {
          switch(Menu)
          {
```

```
        case 1 :
        H1min = H1min + valCod;
        break;

        case 2 :
        H2min = H2min + valCod;
        break;

        case 3 :
        H3min = H3min + valCod;
        break;

        case 4 :
        H4min = H4min + valCod;
        break;
      }
    }
    valCod = 0;
    break;

  case 4 :
    lcd.setCursor(14,2);
    lcd.blink();
    delay(100);
    if (EncodeurA || EncodeurB)
    {
      switch(Menu)
      {
        case 1 :
        H1max = H1max + valCod;
        break;

        case 2 :
        H2max = H2max + valCod;
        break;

        case 3 :
        H3max = H3max + valCod;
        break;

        case 4 :
        H4max = H4max + valCod;
        break;
      }
    }
    valCod = 0;
    break;

    case 5 :
    lcd.setCursor(1,3);
    lcd.blink();
    delay(100);
    if (EncodeurA || EncodeurB)
    {
      switch(Menu)
      {
        case 1 :
        h_onTime1 = h_onTime1 + valCod;
        break;

        case 2 :
        h_onTime2 = h_onTime2 + valCod;
        break;
```

```
        case 3 :
        h_onTime3 = h_onTime3 + valCod;
        break;

        case 4 :
        h_onTime3 = h_onTime3 + valCod;
        break;
      }
    }
    valCod = 0;
    break;

    case 6 :
    lcd.setCursor(4,3);
    lcd.blink();
    delay(100);
    if (EncodeurA || EncodeurB)
    {
      switch(Menu)
      {
        case 1 :
        m_onTime1 = m_onTime1 + valCod;
        break;

        case 2 :
        m_onTime2 = m_onTime2 + valCod;
        break;

        case 3 :
        m_onTime3 = m_onTime3 + valCod;
        break;

        case 4 :
        m_onTime4 = m_onTime4 + valCod;
        break;
      }
    }
    valCod = 0;
    break;

    case 7 :
    lcd.setCursor(12,3);
    lcd.blink();
    delay(100);
    if (EncodeurA || EncodeurB)
    {
      switch(Menu)
      {
        case 1 :
        h_offTime1 = h_offTime1 + valCod;
        break;

        case 2 :
        h_offTime2 = h_offTime2 + valCod;
        break;

        case 3 :
        h_offTime3 = h_offTime3 + valCod;
        break;

        case 4 :
        h_offTime4 = h_offTime4 + valCod;
        break;
      }
```

```
        valCod = 0;
        break;

        case 8 :
        lcd.setCursor(15,3);
        lcd.blink();
        delay(100);
        if (EncodeurA || EncodeurB)
        {
          switch(Menu)
          {
            case 1 :
            m_offTime1 = m_offTime1 + valCod;
            break;

            case 2 :
            m_offTime2 = m_offTime2 + valCod;
            break;

            case 3 :
            m_offTime3 = m_offTime3 + valCod;
            break;

            case 4 :
            m_offTime4 = m_offTime4 + valCod;
            break;
          }
        }
        valCod = 0;
        break;
      }
    }
    lcd.clear();
    lcd.noBlink();
    lcd.setCursor(0,0);
    lcd.print(ligne1);
    lcd.setCursor(0,1);
    lcd.print(ligne2);
    lcd.setCursor(0,2);
    lcd.print(ligne3);
    lcd.setCursor(0,3);
    lcd.print(ligne4);
}
//--------------------------------------------------------------------------------
void Regulation()
{
  // Conversion des heures + minutes en millisecondes
  onTime1 = h_onTime1*3600000 + m_onTime1*60000;
  onTime2 = h_onTime2*3600000 + m_onTime2*60000;
  onTime3 = h_onTime3*3600000 + m_onTime3*60000;
  onTime4 = h_onTime4*3600000 + m_onTime4*60000;
  offTime1 = h_offTime1*3600000 + m_offTime1*60000;
  offTime2 = h_offTime2*3600000 + m_offTime2*60000;
  offTime3 = h_offTime3*3600000 + m_offTime3*60000;
  offTime4 = h_offTime4*3600000 + m_offTime4*60000;

  unsigned long currentMillis1 = millis();
  unsigned long currentMillis2 = millis();
  unsigned long currentMillis3 = millis();
  unsigned long currentMillis4 = millis();

    Serial.print(currentMillis1);
    Serial.print(" - ");
```

```
    Serial.println(" => ");

  //Actionne le relais 1 si la température ou l'hygrométrie dépasse l'un des seuil
  if((Tint > T1max) || (Tint < T1min) || (Hint > H1max) || (Hint < H1min))
  {
    digitalWrite (Sortie1, HIGH);
  }
  else
  {
    if(onTime1 != 0 && offTime1 != 0) // Sinon on déclenche la temporisation si besoin
    {
      if((etatRelais1 == HIGH) && (currentMillis1 - previousMillis1 >= onTime1))
        {
          previousMillis1 = currentMillis1;
          etatRelais1 = LOW;
          digitalWrite(Sortie1, etatRelais1);
          Serial.println(" =>onTime1");
        }
        else if ((etatRelais1 == LOW) && (currentMillis1 - previousMillis1 >= offTime1))
        {
          previousMillis1 = currentMillis1;
          etatRelais1 = HIGH;
          digitalWrite(Sortie1, etatRelais1);
          Serial.println(" =>offTime1 ");
        }
    }
    else
    {
      digitalWrite (Sortie1, LOW);
    }
  }


  //Actionne le relais 2 si la température ou l'hygrométrie dépasse l'un des seuil
  if((Tint > T2max) || (Tint < T2min) || (Hint > H2max) || (Hint < H2min))
  {
    digitalWrite (Sortie2, HIGH);
  }
  else
  {
    if(onTime2 != 0 && offTime2 != 0) // Sinon on déclenche la temporisation si besoin
    {
      if((etatRelais2 == HIGH) && (currentMillis2 - previousMillis2 >= onTime2))
        {
          previousMillis2 = currentMillis2;
          etatRelais2 = LOW;
          digitalWrite(Sortie2, etatRelais2);
        }
        else if ((etatRelais2 == LOW) && (currentMillis2 - previousMillis2 >= offTime2))
        {
          previousMillis2 = currentMillis2;
          etatRelais2 = HIGH;
          digitalWrite(Sortie2, etatRelais2);
        }
    }
    else
    {
      digitalWrite (Sortie2, LOW);
    }
  }

  //Actionne le relais 3 si la température ou l'hygrométrie dépasse l'un des seuil
  if((Tint > T3max) || (Tint < T3min) || (Hint > H3max) || (Hint < H3min))
```

```
    digitalWrite (Sortie3, HIGH);
  }
  else
  {
    if(onTime3 != 0 && offTime3 != 0) // Sinon on déclenche la temporisation si besoin
    {
      if((etatRelais3 == HIGH) && (currentMillis3 - previousMillis3 >= onTime3))
        {
          previousMillis3 = currentMillis3;
          etatRelais3 = LOW;
          digitalWrite(Sortie3, etatRelais3);
        }
        else if ((etatRelais3 == LOW) && (currentMillis3 - previousMillis3 >= offTime3))
        {
          previousMillis3 = currentMillis3;
          etatRelais3 = HIGH;
          digitalWrite(Sortie3, etatRelais3);
        }
    }
    else
    {
      digitalWrite (Sortie3, LOW);
    }
  }

  //Actionne le relais 4 si la température ou l'hygrométrie dépasse l'un des seuil
  if((Tint > T4max) || (Tint < T4min) || (Hint > H4max) || (Hint < H4min))
  {
    digitalWrite (Sortie4, HIGH);
  }
  else
  {
    if(onTime4 != 0 && offTime4 != 0) // Sinon on déclenche la temporisation si besoin
    {
      if((etatRelais4 == HIGH) && (currentMillis4 - previousMillis4 >= onTime4))
        {
          previousMillis4 = currentMillis4;
          etatRelais4 = LOW;
          digitalWrite(Sortie4, etatRelais4);
        }
        else if ((etatRelais4 == LOW) && (currentMillis4 - previousMillis4 >= offTime4))
        {
          previousMillis4 = currentMillis4;
          etatRelais4 = HIGH;
          digitalWrite(Sortie4, etatRelais4);
        }
    }
    else
    {
      digitalWrite (Sortie4, LOW);
    }
  }
}
```