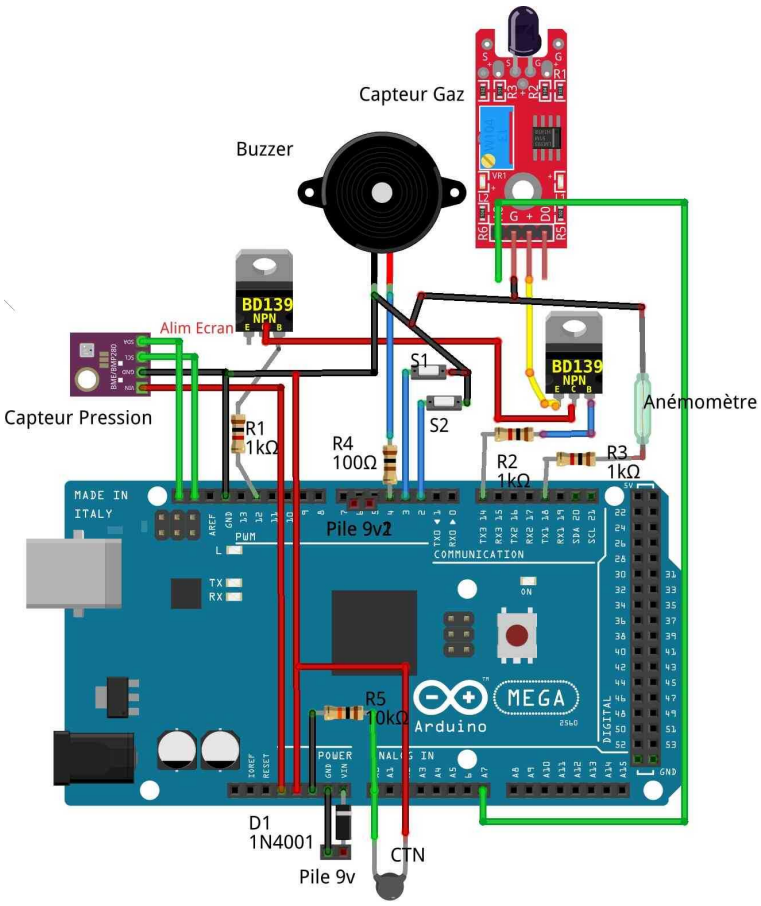
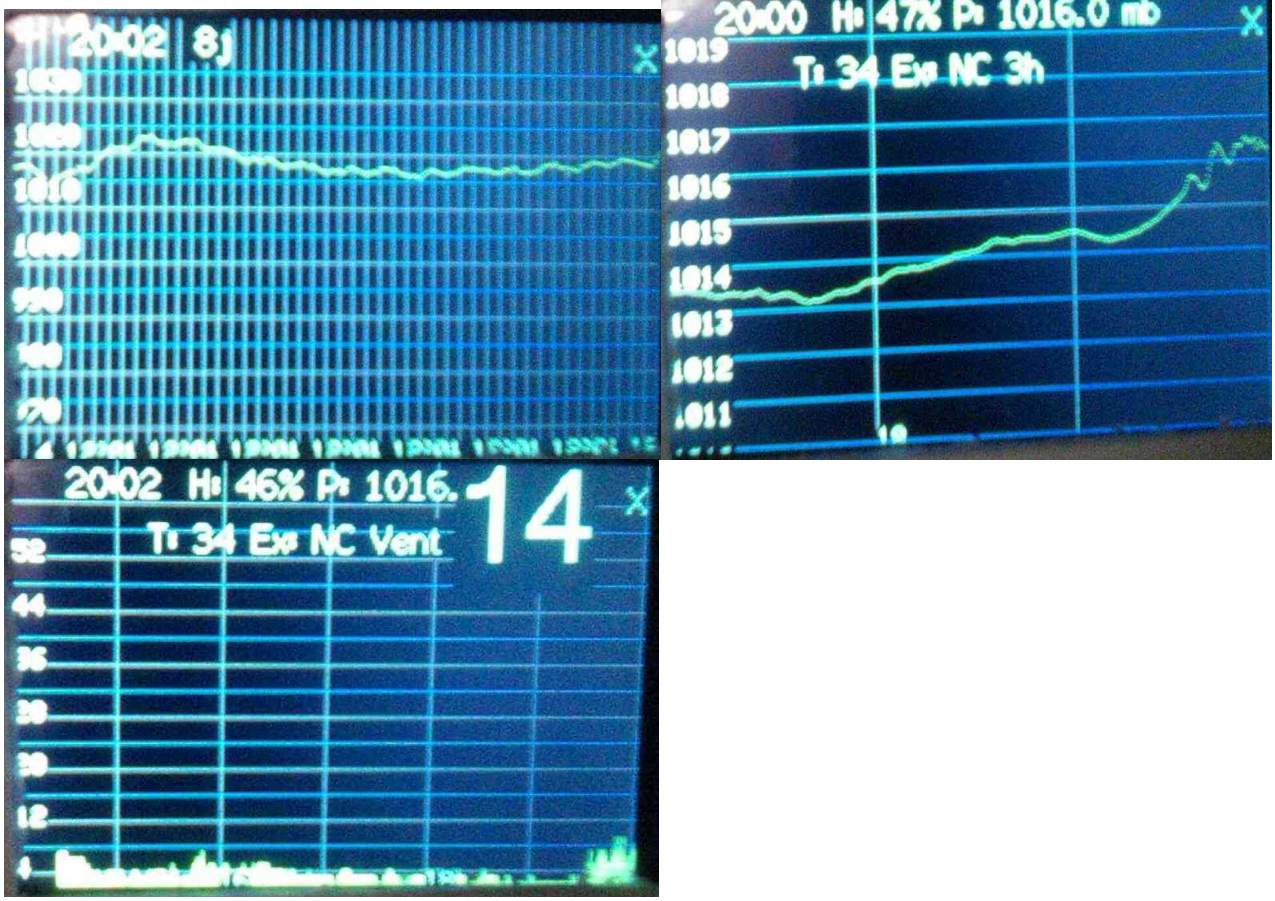

Capteur Gaz
Buzzer
BD139
NPN
Alim Ecran
Capteur Pression
S1
S2
BD139
NPN
Anémomètre
R1
1kΩ
R4
100Ω
R2
1kΩ
R3
1kΩ
MADE IN
ITALY
Pile 9v
COMMUNICATION
MEGA
Arduino
R5
10kΩ
D1
1N4001
Pile 9v
CTN

fritzing


20:00 H: 47% P: 1016.0 mb
T: 34 Ex: NC 3h
1019
1018
1017
1016
1015
1014
1013
1012
1011
20:02 8j
20:02 H: 46% P: 1016.
T: 34 Ex: NC Vent
14

Le programme

```
  display 320*480  jbl 10/2019
  attention: carte mega ou mega 2560
  modif pour blueDot (correction de pression en température)
*/

#include <TFT_HX8357.h> // pour l'écran
#include <Wire.h>
#include <EEPROM.h>
#include "BlueDot_BME280.h"


BlueDot_BME280 Sensor= BlueDot_BME280();
//Sensor Sensor; //Uses I2C address 0x76 (jumper closed)

TFT_HX8357 tft = TFT_HX8357();       // Invoke custom library

//couleurs
#define LTBLUE    0xB6DF
#define LTYELLOW  0xFFF8
#define LTGREY    0xE71C
#define WHITE     0xFFFF
#define BLACK     0x0000
#define RED       0xF800
//ports
#define THERMISTORPIN A0 //analogique pour la température extérieure
#define BoutonG 3 //pour l'affichage
#define BoutonD 2 //réglages
#define AllumeCapteurGaz 14 //met en service le capteur de gaz via un transistor
//#define TestCapteurGaz 15 //la sortie TTL du capteur de gaz (active au niveau bas)
#define TestCapteurGaz A7//analogique pour le gaz (1024 normal. 35 gaz)
#define ecran 12 //allume l'écran via un transistor
#define piezoPin 4 //le buzzeur
#define WindSensorPin (18) //interrupt
byte clic=0;//rotation du capteur 3 par tour
word ventMoy;// sur un intervalle
//les modes d'affichage de la courbe
byte state = 0; // 7 modes respectivement sur 3h, 6h, 12h, 24h, 48h, 96h, 192h
byte heure = 00;
//la donnée de pression est mémorisée à chaque intervalle de 22.5 secondes (3/8 de minute)
//de façon que 480 données (largeur de l'écran) correspondent à 3h (state=0)
//chaque heure contient 160 intervalles
byte interval = 00;
//en principe un intervalle est de 22500 millisecondes.
//Néanmoins, il faut ajuster empiriquement cette valeur en fonction de l'horloge du AT Mega2560
const float delai = 22220;//22297;

//ces deux valeurs comptent respectivement les millisecondes depuis le début
//d'un intervalle de 22.5 secondes et depuis le réveil de l'écran;
//il est important de les déclarer en unsigned long
//http://arlotto.univ-tln.fr/arduino/article/utilisez-correctement-la-fonction
```

```
byte Vent[480];
int indiceVent=0;
byte vitesse;


// la mesure de pression se fait en continu sur chaque intervalle de 22.5s
//on mémorise la moyenne en comptant les relevés
float pression;

//les listes où mémoriser les données de pression
//on mémorise des positions verticales sur l'écran (0 pour 1040mb, 319 pour 960mb)
//on stocke 480 données (largeur de l'écran) pour les 3 premières heures
int Values3[480]; //en 1/100 de mb retiré de 104000 range 0 (1040mb) à 8000 (960 mb)
//pour 6h on prend une donnée sur deux dans Values3 pour les 3 premières heures
//et on a stocké une donnée sur deux pour les 3 heures précédentes dans Valeurs[0]
//pour 2 puissance n *3h on prend une donnée sur deux dans Valeurs[n-1] pour les 2 puissance n-1
*3h
//heures précédentes et on a stocké une donnée sur deux pour les heures précédentes dans Valeurs[n-
1]

int Valeurs[6][240];
//Pour utiliser les listes Values comme des files (piles LIFO) on utilise des index
//qui mémorisent, dans chaque liste, la position de la dernière donnée entrée
word index = 0; //pour Values3 0 à 480
byte indice[6];//pour Valeurs  0 à 240
byte TypeAlarm=2;

//variables pour les alarmes
bool alarmBaro = false; //activée/désactivée
bool alarmGaz = false; //activée/désactivée
bool BaroVent=true;

//variables pour la température extérieure
float ValPortThermistance = 500; //la valeur relévée sur le port THERMISTORPIN
int TempExt = 20; //la température affichée; elle est mémorisée pour le cas où le calcul donne une
valeur aberrante

bool Changestate = false; //gestion rebond
unsigned long somme3;//somme des valeurs pour 3h (mise à jour itérativement)
unsigned long somme6;
unsigned long somme12;
void axes()
{ int i; byte nbPixels;byte InterLigne;
  //une ligne verticale tous les 160 pixels pour state=0 (3h)
  //80 pixels pour state=1 (6h), etc. Correspond aux heures rondes
  // avec étiquette toutes les 2 heures
  if (!BaroVent) nbPixels=2; else nbPixels=pow(2, state);//nb de lignes par unité de temps
  if (nbPixels>16) nbPixels=16;
  InterLigne=160/nbPixels;
//  byte InterLigne[8]={160,80,40,20,10,10,10,80};
  for (i = 479 - interval /nbPixels; i > 0; i -= InterLigne)
```

```
    tft.drawLine(i, 0, i, 319, LTGREY); //tracer
  int h = heure;
  for (i = 479 - interval / nbPixels; i > 0; i -= 2*InterLigne)
  {tft.drawNumber(h, i, 310, 1);
   switch (state) {
    case 5: h = 20 + h; break ;//4 h avant
    case 6: h = 16 + h; break ;//8 h avant
    default: h = 22 + h;break;}//2 h avant
    h = h % 24;}}

//Fonctions pour adresser dans les listes à partir de la dernière donnée entrée.
byte Lvent (int i)
{ int j = indiceVent - i; if (j < 0)j+= 480;
  return Vent[j];}

int LValues3(int i)
{ int j = index - i; if (j < 0)j+= 480;
  return Values3[j];}

int Lvaleurs(byte num, int i)
{ int j = indice[num] - i; if (j < 0)j+= 240;
  return Valeurs[num][j];}

void GradHorizontale(int pas)
{ int i;
  for (i = 0; i < 480; i += pas) tft.drawPixel(479 - i / pas, LValues3(i)/25, LTYELLOW);
  //4 pixels pour 1mb; 1 pixels pour 25/100mb
  //une ligne horizontale tous les 20 pixels (soit 5mb)
  //grise une sur deux0
  //avec graduation à gauche
  for ( i = 39; i <= 299; i += 20) //pas de première ligne
    {if ((i-19)%40 == 0) tft.drawLine(0, i, 479, i, LTGREY);
      else {tft.drawLine(0, i, 479, i, LTBLUE);
            tft.drawNumber(1040 - (i + 1) / 4, 0, i, 1);}}}


void echelle(int pressMax, byte pas)
 {for ( int i = 31; i <320; i += 32) //10 lignes numérotées à partir de Press
     {if (i == 159) tft.drawLine(0, i, 479, i, LTGREY);
      else tft.drawLine(0, i, 479, i, LTBLUE);
     tft.drawNumber(pressMax, 0, i, 1);pressMax-=pas;}}//i+18 décalage ?

void afficheVitesse()
{tft.setCursor(330, 0, 6);//affichage vent
     if (vitesse<10) tft.print("0");tft.print(vitesse);}

void TraceVent() //tracé 6h
  {  RempliTFT();
    for (int i = 0; i < 448; i += 1) // un kn = 5 pixels laisser 32 pixels pour la graduation
    tft.drawLine(479 - i, 319,479 - i,319-5*Lvent(i) , LTYELLOW);
    axes();tft.print(" Vent");  afficheVitesse();
   //5 pixels pour 1kn;
```

```
  //une ligne horizontale tous les 20 pixels (soit 4 kn)
  //grise une sur deux
  //avec graduation à gauche
  for (int i = 0; i <60; i += 4) //pas de première ligne
    {if (i%8 == 0) tft.drawLine(0, 310-5*i, 479, 310-5*i, LTGREY);
      else {tft.drawLine(0, 310-5*i, 479, 310-5*i, LTBLUE);
           tft.drawNumber(i, 0, 319-5*i, 1);}}}


//procédures de tracé paramétriques à partir des listes Values3
//celle-ci va tracer les données des 3 dernières heures,
//avec un pas de 1 pour 3h, de 2 pour 6h, etc.
void trace3Zoom() //tracé 3h sur 10mb
{ int i; int moy=somme3/480;
  int Press=1040-moy+4.5;//pression maxi somme3 somme 480 1/100 de mb
  echelle(Press,1);
   for (i = 0; i < 480; i += 1) // un mb = 36 pixels
    tft.drawPixel(479 - i, map(LValues3(i),moy*100-500,moy*100+500,0,359)-24, LTYELLOW);}

void trace6Zoom() //tracé 6h sur 20mb
{ int i; int moy=(somme3/480+somme6/240)/2;
  int Press=1040-moy+9;//pression maxi somme3 somme 480 1/100 de mb
  echelle(Press,2);
  for (i = 0; i < 240; i += 1)
    tft.drawPixel(239-i, map(Lvaleurs(0,i),moy*100-1000,moy*100+1000,0,359), LTYELLOW);
  for (i = 0; i < 480; i += 2) // un mb = 36 pixels
    tft.drawPixel(479 - i /2, map(LValues3(i),moy*100-1000,moy*100+1000,0,359), LTYELLOW);}

void trace12Zoom() //tracé 6h sur 20mb
{ int i; int moy=(somme3/480+somme6/240+somme12/240)/3;
  int Press=1040-moy+19;//pression maxi somme3 somme 480 1/100 de mb
  echelle(Press,4);
  for (i = 0; i < 240; i += 1)
    tft.drawPixel(239-i, map(Lvaleurs(1,i),moy*100-2000,moy*100+2000,0,359), LTYELLOW);
 for (i = 0; i < 240; i += 2)
    tft.drawPixel(359-i/2, map(Lvaleurs(0,i),moy*100-2000,moy*100+2000,0,359), LTYELLOW);
  for (i = 0; i < 480; i += 4) // un mb = 36 pixels
    tft.drawPixel(479 - i /4, map(LValues3(i),moy*100-2000,moy*100+2000,0,359), LTYELLOW);}

//celle-ci va tracer les données des heures précédentes (num),
//avec un pas qui dépend de state
void tracePlus(byte num,int pas)
{for ( int i = 0; i < 240; i += pas)
     tft.drawPixel(479 - 240 / pas - i / pas, Lvaleurs(num,i)/25, LTYELLOW);}

//la procédure de tracé qui dépend de l'état (3h, 6h, 12h, 24h, 48h.....)
// la tft n'aime pas les procédures récursives d'où la boucle for
void TraceBaro()
 {RempliTFT();axes();
  String Duree[7] = {" 3h", " 6h", " 12h", " 1j", " 2j"," 4j"," 8j"}; //pour affichage
  tft.print(Duree[state]);
  switch(state)
```

```
   {case 0: {trace3Zoom();break;}
    case 1: {trace6Zoom();break;}
    case 2: {trace12Zoom();break;}
 //   case 7: {traceVent();break;}
    default: {byte pas=1;for(byte i=state;i>0;i--) {tracePlus(i-1, pas);pas=pas*2;}
            GradHorizontale(pas);}}}

//les procédures pour l'empilement des données dans les listes Values3
// decale(num)empile une fois sur deux la donnée la plus ancienne de Valeurs[num] (ou Valeurs
pour num=0)
//après avoir appelé decale(num+1) (jusqu'à 5)
bool Bshift[6]; bool Round[6];

void decale(byte num)
{int v;
if (Bshift[num]) {indice[num] =indice[num] +1;
    if (indice[num] == 240){indice[num] = 0;Round[num]=!Round[num];}
   if (num<5)decale(num+1);
   if (num==0)
       {Vent[indiceVent++]=ventMoy/18; ventMoy=0; if (indiceVent==480) indiceVent=0
;//enregistrement vent tous les 2 intervalles
        somme6-=Valeurs[0][indice[0]]/100;v = Values3[index]; somme6+=v/100;}
    else if (num==1) {somme12-=(Valeurs[1][indice[1]])/100;v=Valeurs[0][indice[0]];
somme12+=v/100;} //6h
          else v =Valeurs[num-1][indice[num-1]];
    Valeurs[num][indice[num]]=v;
    word indexEprom=2*(indice[num]+480+num*240);
    if (Round[num]) EEPROM.put(indexEprom,v);
      else EEPROM.put(indexEprom,-v);}
  Bshift[num] = !Bshift[num];}

bool Round3;
//la procédure appelée à chaque intervalle de temps (22.5s)
//elle calcule la donnée correspondant à la pression moyenne sur l'intervalle et l'empile dans Values3
//et teste le gap barométrique
void shift3()
{ if (index++ == 479){index = 0;Round3=!Round3;}//on repart sur 3h, ça permet de trouver index
  decale(0);
  somme3-=Values3[index]/100;//mise à jour pour calcul de pression moyenne sur 3h
  Values3[index] = int(104000 - pression);// / 25);
  somme3+=Values3[index]/100;
  if (Round3) EEPROM.put(2*index,Values3[index]); else EEPROM.put(2*index,-Values3[index]);
  //recherche d'une différence supérieure égale à 4 (1mb tous les 1h)
  alarmBaro = false;
  int Duree[3]={0,160,480};//0 , 1h, 2h
  word gap[3]={0,100,300};// 0, 1 , 3 mb
  for (int i = 1; i < Duree[TypeAlarm]; i++)
    if (abs(Values3[index] - LValues3(i)) > gap[TypeAlarm])
        {alarmBaro = true;break;}}

void RempliTFT() {
  //à partir de la valeur de la thermistance, la température se calcule avec la formule de steinhart
```

```
  //dans cette formule, les constantes A, B et C sont déterminées empiriquement
  //à partir de 3 couples (résistance, température)
  //voir https://en.wikipedia.org/wiki/Steinhart%E2%80%93Hart_equation
  const float A = 0.782302003; //-1.990861069;//multiplié par 1000
  const float B = 0.2648965; //0.697499306;
  const float C = 0.000176909; //-0.001344519;
  //la température donnée par la formule
  tft.setRotation(1);
  tft.setTextColor(WHITE, BLACK);
  tft.fillScreen(BLACK);
   tft.setTextSize(2);
    affiche_heure();
    affiche_alarm();
    tft.setCursor(120,0,2);
    if (state <3){
    tft.print(" H: ");
    tft.print(Sensor.readHumidity(), 0);
    tft.print("% P: ");
    int v=10400 - LValues3(0) / 10;
    tft.print(v/10); tft.print("."); tft.print(v%10);tft.print(" mb");
    tft.setCursor(100,40,2);
    tft.print("T: ");
    tft.print(Sensor.readTempC(), 0);
    //calcul de la résistance R en Ohm R0/(R+R0)=ValPort/1024, avec R0=1kOhm
    float ValThermistance = 10000 * (1024 / (1024 - ValPortThermistance) - 1);
    //calcul de la température par la formule de steinhart
    float steinhart = 1000 / (A + B * log(ValThermistance) + C * pow(log(ValThermistance), 3)) -
273.15;
    if (not isnan(steinhart)) TempExt = int(steinhart); //valeurs aberrantes quand on relance l'écran
    tft.print(" Ex: ");
    if ((TempExt>-10)&&(TempExt<50))tft.print(TempExt); else tft.print("NC");}
//    axes();
}

void GetIndice(byte num)
{int v;bool sgn; word indexEprom=2*(480+num*240);
   if (num==0) somme6=0;else if (num==1) somme12=0;
   for(byte i=0;i<240;i++) {EEPROM.get(indexEprom+2*i,v);
                        if ((!sgn && (v>0)) or (sgn && (v<=0)))
                           {indice[num]=i;Round[num]=sgn;}//repartir sur le même signe
                        sgn=(v>0);if(!sgn)v=-v;Valeurs[num][i]=v;
                        if (num==0)somme6+=v/100;else if (num==1)somme12+=v/100;}}
void GetIndex()
{int v;bool sgn;
  somme3=0;
  for(int i=0;i<480;i++) {EEPROM.get(2*i,v);
                        if ((!sgn && (v>0)) or (sgn && (v<=0)))
                           {index=i;Round3=sgn;} //repartir sur le même signe
                        sgn=(v>0);if(!sgn)v=-v;Values3[i]=v;somme3+=v/100;}}//if (v<180)
v=25*v;provisoire
```

```
void setup()
 {pinMode(BoutonG, INPUT_PULLUP);
  pinMode(BoutonD, INPUT_PULLUP);
  pinMode(ecran, OUTPUT);
  pinMode(AllumeCapteurGaz, OUTPUT);
 // pinMode(TestCapteurGaz, INPUT_PULLUP);
  pinMode(WindSensorPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(WindSensorPin), isr_rotation, FALLING);
  sei();
  //Serial.begin(9600);
  Sensor.parameter.I2CAddress = 0x76;
  Sensor.parameter.sensorMode = 0b11;
  Sensor.parameter.IIRfilter = 0b100;
  Sensor.parameter.humidOversampling = 0b101;
  Sensor.parameter.tempOversampling = 0b101;
  Sensor.parameter.pressOversampling = 0b101;
  Sensor.parameter.tempOutsideCelsius = 15;
  digitalWrite(ecran, HIGH); // pour l'allumage de l'écran
  tft.begin();
  if (Sensor.init() != 0x60) tft.print("Sensor non connecté");
  for (int i=0;i<479;i++) Vent[i]=25*(1+sin(i/60));
  GetIndex();for (byte j=0; j<6;j++) GetIndice(j);}//lecture EEPROM

void maj_heure()
 {if (interval++ == 159) // il  faut 160 intervalles pour 1h
  {interval = 0;if (heure++ == 23)  heure = 0;}}

void affiche_heure()
 {tft.setCursor(40, 0, 2);
  if (heure < 10) tft.print("0");
  tft.print(heure);
  if (interval < 26) tft.print(":0");
  else tft.print(":");
  tft.print(interval * 3 / 8);}

void prec_heure()
 {if (interval-- == 0) // il  faut 160 intervalles pour 1h
  {interval = 159;if (heure-- == 0)  heure = 23;}}// c'est des bytes

void affiche_alarm()
 {//tft.setTextSize(2);
  tft.setCursor(450, 10,2);
  if (alarmBaro)  tft.setTextColor(WHITE, RED);
  switch (TypeAlarm){
   case 0:{tft.print(" X"); break;}
   case 1:{tft.print(" T"); break;}//thunderstorm
   case 2:{tft.print(" S"); break;}}}// storm

// This is the function that the interrupt calls to increment the rotation count
void isr_rotation ()
   {unsigned long ContactBounceTime;
    if ((millis() - ContactBounceTime) > 5 )   // debounce the switch contact.
```

```
      { clic++; ContactBounceTime = millis();}}

void loop(void) {
  unsigned long TempsReveil;
  String message = "";
  if (digitalRead(ecran) == HIGH)
    if (BaroVent) TraceBaro(); else TraceVent();
  if (alarmBaro) tone(piezoPin, 1000, 1000);//une seconde toutes les 3/8 de mn
  if (alarmGaz ) tone(piezoPin, 6000, 3000);// trois secondes pour le gaz plus aigu && AlarmOn (ne
jamais inhiber l'alrme gaz)
  unsigned long temps = millis(); //Important pour stocker les millis
  byte nbReleves = 1;//pour calcul de moyenne sur l'intervalles
  //boucle pour un intervalle de 22.5 s
//  ???ventMoy=0;cli();clic=0;sei();//effacer clics pendant réaffichage Attention la tft a besoin des
interruptions!
  while (millis() - temps <= delai) // syntaxe pour tenir compte de ce que millis retourne à zéro après
50jours !
   {if (millis() - temps > nbReleves * 1000) //on fait une mesure toutes les secondes pour moyenner
     {if ((nbReleves % 3)==0)// toutes les 3 secondes pour le vent
       {cli();vitesse=clic/2;clic=0;sei(); //Disable interrupts, calcule vitesse et Enables interrupts
         if (!BaroVent) afficheVitesse();
//         if (vitesse>ventMoy)
        ventMoy+=vitesse;}// vitesse max sur 2 intervalles (55s)
       pression += (Sensor.readPressure()*100 - pression) / nbReleves; //moyennes calculées
itérativement pour éviter dépassement
       ValPortThermistance += (analogRead(THERMISTORPIN) - ValPortThermistance) /
nbReleves++;}

    if (alarmBaro && (digitalRead(BoutonG) == LOW)) TypeAlarm = 0; //pour stopper l'alarme à
remettre par l'utilisateur
    if (alarmGaz && (digitalRead(BoutonG) == LOW)) alarmGaz = false; //pour stopper l'alarme
    // réveil de l'écran
    if ((digitalRead(ecran) == LOW) && (digitalRead(BoutonG) == LOW))
     {digitalWrite(ecran, HIGH);;BaroVent=true;
      tft.begin();TraceBaro(); TempsReveil = millis();}

    //changement de mode et réaffichage dans le cas où l'écran est réveillé
    static byte delaiBoutonG;//pour mise à l'heure croissant
    if ((digitalRead(ecran) == HIGH) && (digitalRead(BoutonG) == LOW))
     if (BaroVent)
      {delay(delaiBoutonG);
       if (digitalRead(BoutonG) ==HIGH) Changestate = true; //appui court
        else {prec_heure();affiche_heure();if (delaiBoutonG-- < 5)delaiBoutonG = 5;}}
      else {TraceBaro();BaroVent=true;TempsReveil = millis();}
       else delaiBoutonG = 200;
    //délai de 0.2 s pour éviter un rebond
    if (Changestate)
     {Changestate = false;
      if (state++ == 6) state = 0;//cyclage dans les écrans baro
      TraceBaro();}
```

```
    //extinction de l'écran au bout de 40s sauf si appui BoutonD ou Vent
    if ((digitalRead(ecran) == HIGH) && (digitalRead(BoutonD) == HIGH)
    &&((millis() - TempsReveil > 40000) and (BaroVent)))
      digitalWrite(ecran, LOW);
    //test de l'alarme de gaz
    if ((digitalRead(AllumeCapteurGaz) == HIGH) and (analogRead(TestCapteurGaz)<50))
alarmGaz = true;
    static byte delaiBoutonD;//pour mise à l'heure
    if (digitalRead(BoutonD) == LOW)
      if (digitalRead(ecran) == HIGH) //écran allumé -> réglages
        {delay(delaiBoutonD);
         if (digitalRead(BoutonD) == HIGH) //appui court->toggle alarm ou barovent
          if (BaroVent)
             {BaroVent=false;TraceVent();}
          else
             {if (TypeAlarm++==2) TypeAlarm=0;
                affiche_alarm();}
         else
          {maj_heure(); //appui long->incrémente minute
           affiche_heure();
           if (delaiBoutonD-- < 5)delaiBoutonD = 5;}}
      else {digitalWrite(ecran, HIGH);
            tft.begin();BaroVent=false;TraceVent();}
    else delaiBoutonD = 200;}


  //à l'issue de l'intervalle
   if (interval % 16 == 0) digitalWrite(AllumeCapteurGaz, HIGH);
  else digitalWrite(AllumeCapteurGaz, LOW);
  //mise à jour de l'heure
   maj_heure();
  shift3();
}//empilage de la pression
```