THERMOWEB UN THERMOSTAT CONNECTE AVEC ARDUINO

REALISER SON THERMOSTAT CONNECTE

Les thermostats connectés du commerce sont très couteux et souvent associés à une offre de service global. Grâce à arduino nous allons réaliser un thermostat connecté à partir d'une carte méga 2560, d'un shield internet et de quelques composants électroniques.

Facade du thermostat

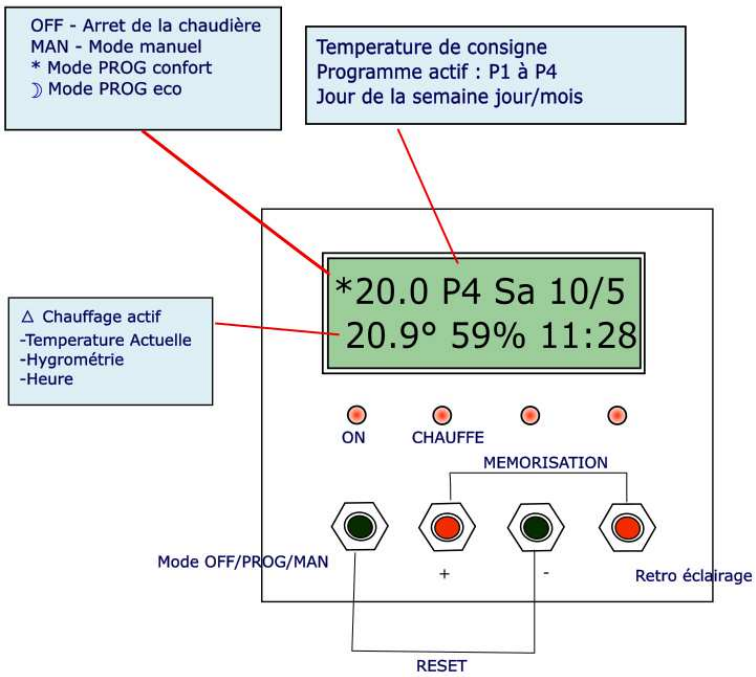
CARACTERISTIQUES DE THERMOWEB

- 3 modes (programmation, manuel, arrêt)
- 3 températures de consigne (éco, confort, manuel)
- 4 programmes journaliers différents à affecter au jour de la semaine. Un programme permet d'affecter par demi-heure (de oh à 23h30) une température de consigne éco ou confort.
- En cas de coupure électrique reprise de la situation mémorisée, température éco, confort, manuel, ainsi que le mode (programmation, manuel ou arrêt).
- Gestion automatique de l'heure et du calendrier avec prise en compte du changement d'heure été/hivers et synchronisation avec un serveur NTP.
- Programmation d'un reset automatique quotidien à 2h00 et 3h00 du matin pour mettre à jour l'heure depuis un serveur NTP et de caler heure d'hivers et heure d'été lorsqu'il y a un changement.
- Connexion http sécurisé par mot de passe pour accéder à la page web du thermostat. Par défaut il n'est possible de voir que l'état du thermostat, une fois authentifié les boutons apparaissent et permettent de changer le mode, les températures de consignes, de mémoriser ou d'effectuer un reset.
- Un affichage sur écran LCD : date, heure, température et taux d'humidité, le mode, la période (éco, confort), la température de consigne.
- Une option facultative. Contrôler l'état d'un contacteur et faire clignoter le rétroéclairage du thermostat pour attirer l'attention.

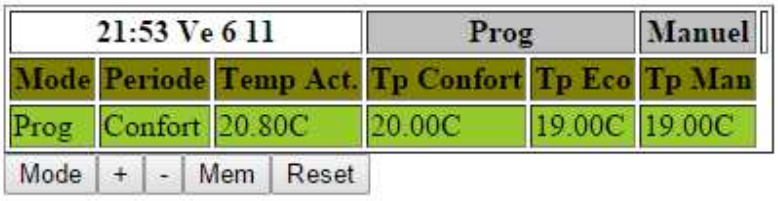
« Un projet développé pendant un an et qui gère mon installation de chauffage depuis 3 ans. »

UN THEMOSTAT QUI SE PILOTE EN LOCAL

Grâce à ses quatre boutons poussoirs il est possible de changer le mode de fonctionnement du thermostat, de varier la température de consigne, de mémoriser la situation de reprise en cas de coupure électrique, de reseter le thermostat et d'activer le rétro-éclairage de l'écran lcd.



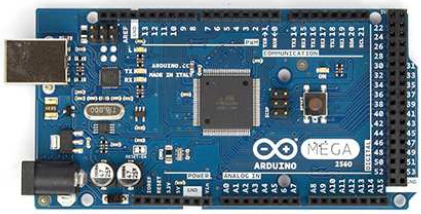
UN THEMOSTAT QUI SE PILOTE VIA INTERNET



En se connectant à la page web du thermostat, il est possible d'agir sur celui de la même manière qu'en mode local. On retrouve toutes les fonctionnalités changement de mode de fonctionnement, variation de température de consigne, etc…

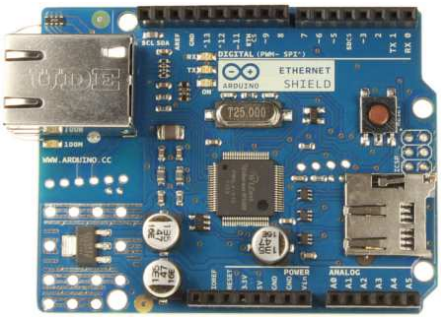
COTE MATERIEL

UN ARDUINO MEGA 2560



http://arduino.cc/en/Main/arduinoBoardMega2560

UNE CARTE ETHERNETSHIELD

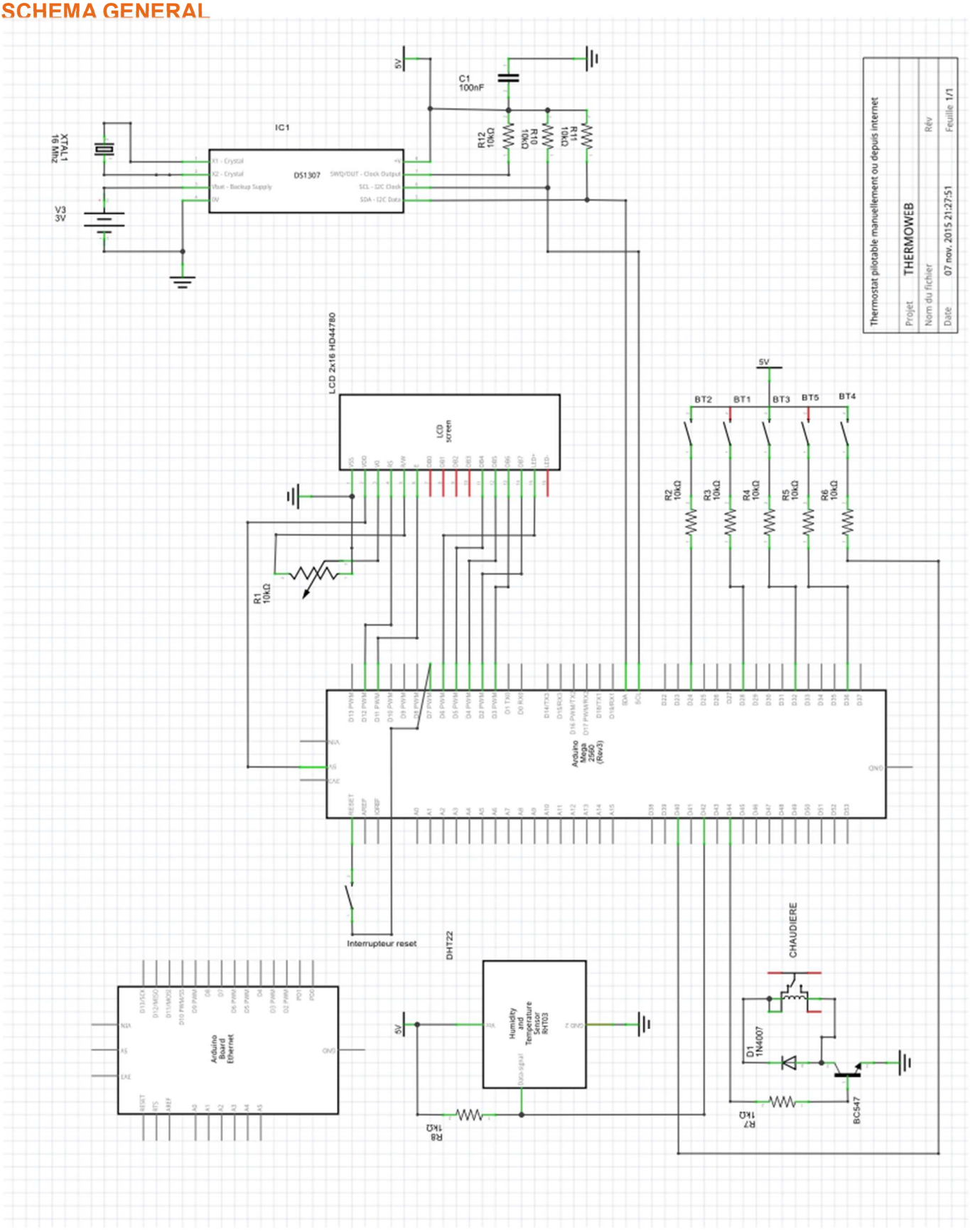


http://arduino.cc/en/Main/ArduinoEthernetShield

COTE COMPOSANTS

- Un afficheur rétroéclairé 2x16 caractères Noirs /Jaune-Vert de type HD44780
- Un module horloge DS1307
- Un module température et hygrométrie : DHT22
- Des leds
- Des boutons poussoirs
- Des connecteurs modulaires 2,54mm Mâle
- Des connecteurs modulaires 2,54mm Femelle vide
- Un module relai pour piloter la chaudière composé de :
    - Une diode 1N4007
    - Un Transistors bipolaire NPN - BC547B
    - Un Relais de puissance 6V 1RT 16A HF115F
- Une plaque d'essai à bande
- Un interrupteur pour gérer le reset logique de l'arduino

SCHEMA GENERAL



Mise à jour du programme journalier du thermostat

Quelques éléments sont à mettre à jour directement dans le programme. L'adresse IP, le détail des programmes P1 à P4, le programme journalier PS et les seuils de température maximum souhaités.

Mise à jour des programmes

- 4 programmes journaliers : PR1, PR2, PR3, PR4. Chaque programme à une plage de 0h00 à 23h30 avec des valeurs par demi-heure de 0 ou 1 correspondant à 0 : température éco , 1 : température confort. Cette programmation permet de définir la température de consigne par demi-heure, la première valeur définit la plage de 0h00 à 0h30, la suivante de 0h30 à 1h00 …. et la dernière de 23h30 à 0h00.

- Une programmation journalière : PS contenant la programmation du Dimanche au Samedi, c'est-à-dire on affecte au jour de la semaine le programme adapté, 1, 2,3 ou 4, correspondant à PR1, PR2, PR3 et PR4.

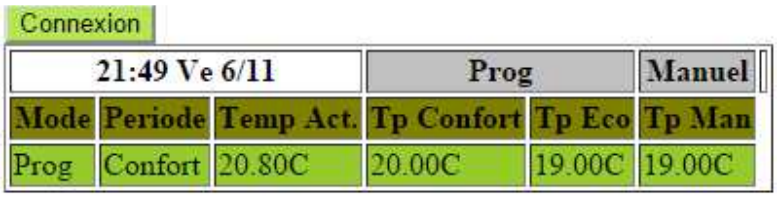
```
//Quatre programme de chauffage journalier par jour/heure de 0 à 23h30 et par demi-heure

const prog_uint16_t PR1[] PROGMEM =
{0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0};
const prog_uint16_t PR2[] PROGMEM =
{0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0};
const prog_uint16_t PR3[] PROGMEM =
{0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0};
const prog_uint16_t PR4[] PROGMEM =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0};

const uint16_t PS[] PROGMEM = {4,1,1,3,2,2,4}; //D, L, M, M, J, V, S
```

Mise à jour des paramètres de la carte réseau de l'arduino

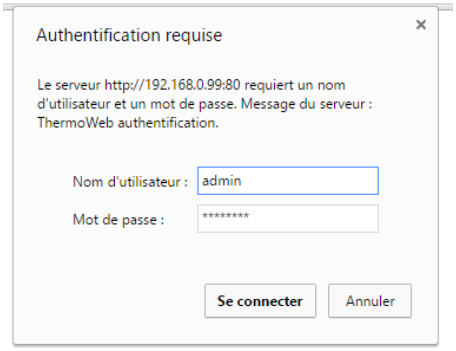
```
//Declaration pour module Ethernet
static uint8_t mac[] = { 0xAA, 0xAA, 0xAA, 0xEF, 0xFE, 0xEF };
static uint8_t ip[] = { 192, 168, 0, 10 }; // adresse IP de la carte ethernet
unsigned int localPort = 8888;      // local port to listen for UDP packets
IPAddress timeServer(212, 37, 192, 31); // adresse IP serveur NTP en France
```

Mettre à jour la ligne en rouge ci-dessus, correspondant à l'adresse IP de la carte EthernetShield en fonction de votre réseau.

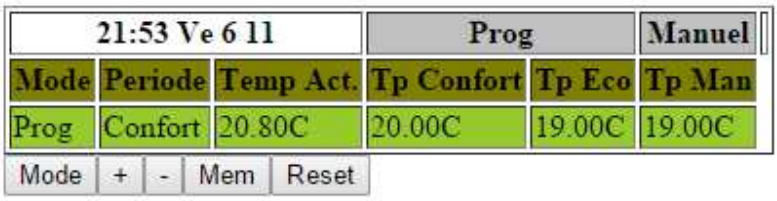
L'accès au thermostat depuis le web est sécurisé par un login et un mot de passe. A la connexion seul les paramètres du thermostat apparaissent.



Cependant en cliquant sur connexion, une fenêtre d'authentification apparaît pour vous invitez à nous authentifier.



Après la saisie du nom d'utilisateur (login) et du mot de passe les boutons permettant de modifier l'état du thermostat apparaissent.



Le login et le mot de passe sont codifiés en base 64.

Pour vous aider à faire cette codification vous pouvez vous rendre sur ce site :
http://www.opinionatedgeek.com/DotNet/Tools/Base64Encode/default.aspx

Le champ à convertir est au format suivant « login:password ».

Si nous avons par exemple, pour le login : **admin** et le mot de passe : **arduino**, il faut encoder en base64 : **admin:arduino**, ce qui donne en conversion base 64 : **YWRtaW46YXJkdWlubw==**

Il faut donc modifier la ligne suivante du programme

```
if (server.checkCredentials("YWRtaW46YXJkdWlubw=="))
```

Mise à jour des seuils de température

```
//Bornage des températures de consigne
 if (Tc > 22) Tc=22;
 if (Tm > 22) Tm=22;
 if (Te > 19) Te=19;
```

Il est possible de modifier les valeurs maxi des températures de confort, manuel et économique. Il ne sera pas possible alors de dépasser ces valeurs par l'utilisateur. Dans notre cas, La température de confort et la température manuelle ne peuvent pas dépasser 22° et la température économique 19°.

Explications

Configuration des Ports E/S

Avant été confronté à un code mal optimisé il a fallu trouver un certain nombre d'astuce pour que le code fonctionne sans anomalie. L'utilisation des registres pour la gestion des broches a été nécessaire.
Cela consiste à remplacer les commandes, pinMode, digitalWrite, digitalRead() par des commandes registres.

- pinMode(26, OUTPUT); devient DDRA |= (1<<4); //LED1
- digitalWrite(26, LOW); devient PORTA &= ~(1<<4);
- digitalWrite(26, HIGH); devient PORTA |= (1<<4);

DDRx permet de définir une broche en une entrée ou une sortie (input 0, output 1)
PORTx pour modifier l'état de la broche (high ou low)
PINx pour lire l'état d'une broche

Pour l'arduino mega 2560 vous pouvez vous aider du tableau ci-dessous pour retrouver la correspondance entre la codification des registres et les numéros des broches.

PA	7	6	5	4	3	2	1	0
D22								X
D23							X	
D24						X		
D25					X			
D26				X				
D27			X					
D28		X						
D29	X							

PB	7	6	5	4	3	2	1	0
D53								X
D52							X	
D51						X		
D50					X			
D10				X				
D11			X					
D12		X						
D13	X							

PC	7	6	5	4	3	2	1	0
D37								X
D36							X	
D35						X		
D34					X			
D33				X				
D32			X					
D31		X						
D30	X							

PD	7	6	5	4	3	2	1	0
D21								X
D20							X	
D19						X		
D18					X			
				X				
			X					
		X						
D38	X							

PE	7	6	5	4	3	2	1	0
D0								X
D1							X	
						X		
D5					X			
D2				X				
			X					
		X						
	X							

PF	7	6	5	4	3	2	1	0
A0								X
A1							X	
A2						X		
A3					X			
A4				X				
A5			X					
A6		X						
A7	X							

PG	7	6	5	4	3	2	1	0
D41								X
D40							X	
D39						X		
					X			
				X				
D4			X					
		X						
	X							

PH	7	6	5	4	3	2	1	0
D17								X
D16							X	
						X		
D6					X			
D7				X				
D8			X					
D9		X						
	X							

PJ	7	6	5	4	3	2	1	0
D15								X
D14							X	
						X		
					X			
				X				
			X					
		X						
	X							

PK	7	6	5	4	3	2	1	0
A8								X
A9							X	
A10						X		
A11					X			
A12				X				
A13			X					
A14		X						
A15	X							

PL	7	6	5	4	3	2	1	0
D49								X
D48							X	
D47						X		
D46					X			
D45				X				
D44			X					
D43		X						
D42	X							

Ex :

Le code suivant correspond à la led 1, mais à quel port de l'arduino : DDRA |= (1<<4); //LED1

DDR**A** |= (1<<**4**); //LED1 => A 4 (tableau PA, la colonne 4, intersection du X coincide avec D26

Affectation des broches pour le projet :

```
DDRA |= (1<<4); //LED1
DDRA |= (1<<0); //LED2
DDRC |= (1<<7); //LED3 non utilisée
DDRD |= (1<<7); //LED4 non utilisée
DDRH |= (1<<3); //LED+
DDRL |= (1<<5); //Relais Chauffage
DDRA |= (0<<6); //BT1 - changement mode
DDRA |= (0<<2); //BT2 décrément température de consigne
DDRC |= (0<<5); //BT3 incrément température de consigne
DDRG |= (0<<1); //BT4 illumination LCD
DDRC |= (0<<1); //BT5 capteur état porte de garage
PORTH |= 1<<4; //Indispensable
DDRH |= (1<<4); // PIN 7 output (reset arduino)

  //DS1307
DDRE |= (1<<4); // Test of the SQW pin, D2 = INPUT
PORTE |= 1<<4; // Test of the SQW pin, D2 = Pullup on
```

Hors codification registre

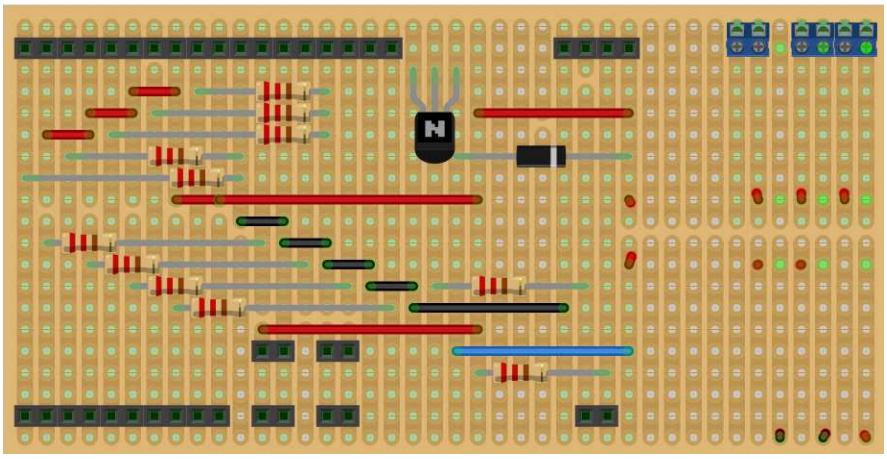
```
// Declaration de la sonde de temperature et d'humidité DHT22
#define DHTPIN  42
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

//Declaration pour écran LCD 2x16
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

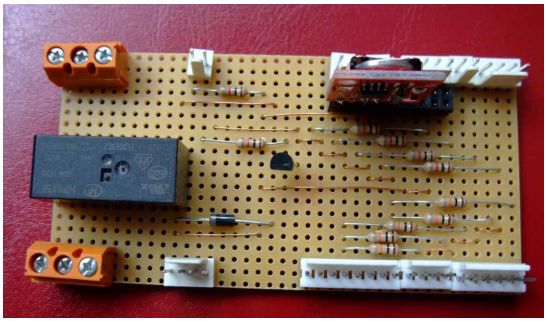
Si vous souhaiter faire évoluer ce programme il faudra faire attention de ne pas utiliser la broche D4, utilisée par l'ethernetshield pour la carte SD. Cela risque d'engendrer un dysfonctionnement.

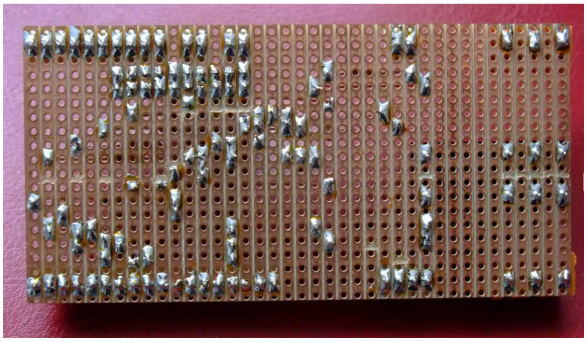
A savoir :

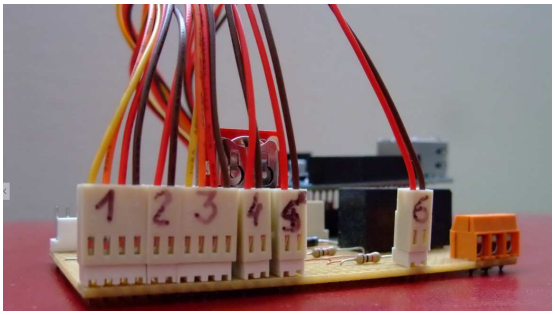
Le port D7 est connecté au port reset afin de permettre un reset logique en agissant sur le port D7. Cependant cette connexion pose parfois un problème de téléchargement d'un nouveau microcode sur l'arduino mega. Pour éviter d'avoir à retirer cette connexion un interrupteur est utilisé et doit rester en position fermé en fonctionnement normal et ouvert en mode téléchargement.

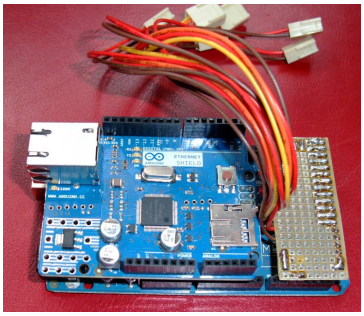
LE CABLAGE

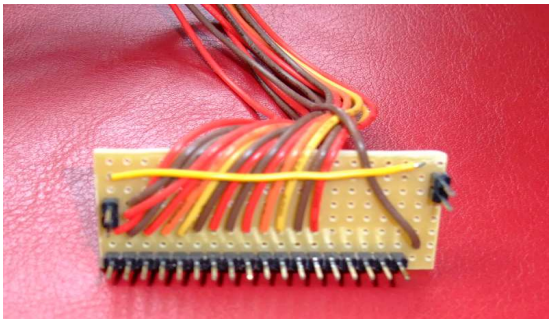












LE CODE COMPLET

```
#define WEBDUINO_AUTH_REALM "ThermoWeb authentification"
#include <avr/pgmspace.h>
#include <SPI.h>
#include <Ethernet.h>
#include <WebServer.h>
#include <EthernetUdp.h>
#include <Wire.h>                  // For some strange reasons, Wire.h must be included here
#include <RTClib.h>
#include <LiquidCrystal.h>
#include <EEPROM.h>
#include "DHT.h"

// Declaration sonde temperature et humidite
#define DHTPIN  42
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);


//Encapsulation RTC pour module horloge
RTC_DS1307 RTC;

//Declaration pour module Ethernet
static uint8_t mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
static uint8_t ip[] = { 192, 168, 0, 99 };
unsigned int localPort = 8888;      // local port to listen for UDP packets
IPAddress timeServer(212, 37, 192, 31); // serveur NTP en France
const int NTP_PACKET_SIZE= 48; // NTP time stamp is in the first 48 bytes of the message
byte packetBuffer[ NTP_PACKET_SIZE]; //buffer to hold incoming and outgoing packets
// A UDP instance to let us send and receive packets over UDP
EthernetUDP Udp;
unsigned long epoch; //Unix Epoch time (NTP or RTC depending on state)


//Declaration pour ecran LCD 2x16
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
uint8_t lune[8] = {0x00,0x07,0x0E,0x0E,0x0E,0x0E,0x07,0x00};
uint8_t sole[8] = {0x00,0x15,0x0E,0x1B,0x0E,0x15,0x00,0x00};
uint8_t P1[8] = {0x18,0x14,0x14,0x1A,0x16,0x12,0x12,0x07};
uint8_t P2[8] = {0x18,0x14,0x14,0x1A,0x15,0x11,0x12,0x07};
uint8_t P3[8] = {0x18,0x14,0x14,0x1B,0x11,0x13,0x11,0x03};
uint8_t P4[8] = {0x18,0x14,0x14,0x18,0x14,0x15,0x17,0x01};

// Variables memorisees prise en compte en cas de coupure de courant
float Tc = EEPROM.read(0);    //Temperature de Confort memorisee
float Te = EEPROM.read(1);    //Temperature Eco  memorisee
float Tm = EEPROM.read(2);    //Temperature Manuel memorisee
byte ST1 = EEPROM.read(3);    //Mode thermostat memorisee (Prog, Manuel, Off)

//declaration variables de traitement
float Ta;           //Temperature Actuelle
float DIFF;           //Difference Tempature Actuelle et Temperature de consigne
float t =0.5;         //Increment pour reglage de la temperature de de consigne
float TpCons;         //Temperature de Consigne
float b;
float Hu;           //Taux Humidite Actuel
byte PR;
byte PG;
unsigned long TPCHAUFF;
unsigned long TPARRET;
unsigned long STARTA = 0;
unsigned long STARTC = 0;
unsigned long ACTUELLE;
String Mode = "";       // Mode Chaudiere (A pour Arret, C pour Chauffe)

//variables pour calcul de l heure ete/hivers
uint8_t cpt = 41;
uint8_t d1=31;
uint8_t m1;
uint8_t jdc;
uint8_t jo;
```

```
uint8_t siecle;
uint8_t JCM;
uint8_t JCO;
uint8_t j;
uint8_t m;
uint8_t h;
int y1;
int z;
boolean etatChaud = false;
boolean Cons;
char str[3];

//Quatre programme de chauffage journalier par jour/heure de 0 à 23h30 et par demi heure

const prog_uint16_t PR1[] PROGMEM =
{0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0};
const prog_uint16_t PR2[] PROGMEM =
{0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0};
const prog_uint16_t PR3[] PROGMEM =
{0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0};
const prog_uint16_t PR4[] PROGMEM =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0};
const uint16_t PS[] PROGMEM = {4,2,2,2,2,2,4}; //D, L, M, M, J, V, S affectation des programmes par jour

const prog_char* wd[] = {"Di","Lu","Ma","Me","Je","Ve","Sa"};
const prog_char* md[] = {"Off","Prog","Manuel"};


//Declaration des pages web
P(cmdWeb) =
"<td><input type=\"button\" name=\"nom\" value=\"Mode\" onclick=\"self.location.href='bt1.html'\" onclick></td>"
"<td><input type=\"button\" name=\"nom\" value=\"+\" onclick=\"self.location.href='bt2.html'\" onclick></td>"
"<td><input type=\"button\" name=\"nom\" value=\"-\" onclick=\"self.location.href='bt3.html'\" onclick></td>"
"<td><input type=\"button\" name=\"nom\" value=\"Mem\" onclick=\"self.location.href='mem.html'\" onclick></td>"
"<td><input type=\"button\" name=\"nom\" value=\"Reset\" onclick=\"self.location.href='res.html'\"
onclick></td>"
"</tr>""</table>";

P(tbWeb) =
"<table border=\"1\" >""<tr><th colspan=3>";
P(tb1Web) =
"</th><th colspan=2 bgcolor=\"silver\">Prog</th><th bgcolor=\"silver\">Manuel</th><th></th></tr>"
"<tr bgcolor=\"olive\"><th>Mode</th><th>Periode</th> <th>Temp Act.</th> <th> Tp Confort </th><th> Tp Eco </th>
<th> Tp Man </th></tr>"
"<tr bgcolor=\"#96CA2D\"><td>";


#define PREFIX ""
WebServer webserver(PREFIX, 80);

void defaultCmd(WebServer &server, WebServer::ConnectionType type, char *, bool)
{
  server.httpSuccess();
  if (type != WebServer::HEAD)
  {
    P(Homepage) =
      "<td> <input type=\"submit\" value=\"Connexion\"  style=\"background-color: #B5E655;\"
onclick=\"location=\'private.html\'\"/></td></br>";
    server.printP(Homepage);
    DateTime now = RTC.now();
    char str[3];
    Ta = dht.readTemperature();
    server.printP(tbWeb);
    server.print(format2(str, (now.hour())));
    server.print(':');
    server.print(format2(str, (now.minute())));
    server.print(' ');
    server.print(wd[((now.dayOfWeek()))]);
    server.print(' ');
    server.print((now.day()));
    server.print('/');
    server.print(now.month());
   if ((PINA & (1<<0)) != 0) {
          server.print(" *");
    }
```

```
    server.printP(tb1Web);
    server.print(md[ST1]);
    server.print("</td><td>");
    server.print( (Cons==0) ? "Eco" : "Confort" );
    server.print("</td><td>");
    server.print(Ta);
    server.print("C</td><td>");
    server.print(Tc);
    server.print("C</td><td>");
    server.print(Te);
    server.print("C</td><td>");
    server.print(Tm);
    server.print("C</td></tr></table>");
  }
}
void privateCmd(WebServer &server, WebServer::ConnectionType type, char *, bool)
{
  if (server.checkCredentials("xxxxxxxxxxxxxxxxxxxxxxxxxxxx"))
  {
    server.httpSuccess();
    if (type != WebServer::HEAD)
    {
      DateTime now = RTC.now();
      char str[3];
      Ta = dht.readTemperature();
      server.printP(tbWeb);
      server.print(format2(str, (now.hour())));
      server.print(':');
      server.print(format2(str, (now.minute())));
      server.print(' ');
      server.print(wd[(now.dayOfWeek())]);
      server.print(' ');
      server.print((now.day()));
      server.print(' ');
      server.print(now.month());
     if ((PINA & (1<<0)) != 0) {
      server.print(" *");
    }
      server.printP(tb1Web);
      server.print(md[ST1]);
      server.print("</td>");
      server.print("</td><td>");
      server.print( (Cons==0) ? "Eco" : "Confort" );
      server.print("</td><td>");
      server.print(Ta);
      server.print("C</td><td>");
      server.print(Tc);
      server.print("C</td><td>");
      server.print(Te);
      server.print("C</td><td>");
      server.print(Tm);
      server.print("C</td></tr></table>");
      server.printP(cmdWeb);
    }
  }
  else
  {
    server.httpUnauthorized();
  }
}
void bt1Cmd(WebServer &server, WebServer::ConnectionType type, char *, bool){
  TraitBt(1);
  server.httpSeeOther(PREFIX "/aff.html");
}
void bt2Cmd(WebServer &server, WebServer::ConnectionType type, char *, bool){
  TraitBt(2);
  server.httpSeeOther(PREFIX "/aff.html");
}
void bt3Cmd(WebServer &server, WebServer::ConnectionType type, char *, bool){
  TraitBt(3);
  server.httpSeeOther(PREFIX "/aff.html");
}
void resCmd(WebServer &server, WebServer::ConnectionType type, char *, bool){
  TraitReset();
  server.httpSeeOther(PREFIX "/aff.html");
}
```

```
void memCmd(WebServer &server, WebServer::ConnectionType type, char *, bool){
  Memory();
  server.httpSeeOther(PREFIX "/aff.html");
}
void pageWeb(WebServer &server, WebServer::ConnectionType type, char *, bool){

  DateTime now = RTC.now();
  char str[3];
  Ta = dht.readTemperature();
  server.printP(tbWeb);
  server.print(format2(str, (now.hour())));
  server.print(':');
  server.print(format2(str, (now.minute())));
  server.print(' ');
  server.print(wd[(now.dayOfWeek())]);
  server.print(' ');
  server.print((now.day()));
  server.print(' ');
  server.print(now.month());
  if ((PINA & (1<<0)) != 0) {
      server.print(" *");
    }
  server.printP(tb1Web);
  server.print(md[ST1]);
  server.print("</td>");
  server.print("</td><td>");
  server.print( (Cons==0) ? "Eco" : "Confort" );
  server.print("</td><td>");
  server.print(Ta);
  server.print("C</td><td>");
  server.print(Tc);
  server.print("C</td><td>");
  server.print(Te);
  server.print("C</td><td>");
  server.print(Tm);
  server.print("C</td></tr></table>");
  server.printP(cmdWeb);
}
void setup()
{

  lcd.createChar(0, lune);
  lcd.createChar(1, sole);
  lcd.createChar(2, P1);
  lcd.createChar(3, P2);
  lcd.createChar(4, P3);
  lcd.createChar(5, P4);
  DDRA |= (1<<4); //LED1
  DDRA |= (1<<0); //LED2
  DDRC |= (1<<7); //LED3
  DDRD |= (1<<7); //LED4
  DDRH |= (1<<3); //LED+
  DDRL |= (1<<5); //Relais Chauffage
  DDRA |= (0<<6); //BT1 - changement mode
  DDRA |= (0<<2); //BT2 decrement temperature de consigne
  DDRC |= (0<<5); //BT3 increment temperature de consigne
  DDRG |= (0<<1); //BT4 illumination LCD
  DDRC |= (0<<1); //BT5 capteur porte de garage
  PORTH |= 1<<4;  //We need to set it HIGH immediately on boot
  DDRH |= (1<<4); // PIN 7 output (reset arduino)

   //DS1307

  DDRE |= (1<<4); // Test of the SQW pin, D2 = INPUT
  PORTE |= 1<<4; // Test of the SQW pin, D2 = Pullup on

  Serial.begin(9600);
  lcd.begin(16, 2);
  Wire.begin();
  dht.begin();
  RTC.begin();

  //Mise a jour de l'heure via un serveur NTP
      lcd.setCursor(0, 0);
    lcd.print("Synchronisation");
    lcd.setCursor(0, 1);
```

```
    lcd.print("Heure Internet");
  //Serial.println("Starting Ethernet"); //configure Ethernet
    if(Ethernet.begin(mac) != 0) {
  //Serial.println("Ethernet Configured");
      Udp.begin(localPort);
      //get the NTP timestamp
      epoch = getNTP();
      //set the RTC
      RTC.adjust(epoch);
      DateTime now = RTC.now();
      //Serial.print(now.hour());
      //Serial.print(":");
      //Serial.print(now.minute());
    //L heure a ete recupere sur un serveur NTP
    //Traitement pour calcul heure ete/hivers
    //
    //Calcul quel jour tombe le 31 mars de l annee
      m1 = 3;
      y1 = now.year();
      JDS();
          //Calcul du dimanche de changement d heure
      if (jdc == 6){
        JCM = 31;
      } else {
        JCM = 30 - jdc;
      }
    //Calcul quel jour tombe le 31 octobre de l annee
      m1 = 10;
      JDS();
          //Calcul du dimanche de changement d heure
      if (jdc == 6){
        JCO = 31;
      } else {
        JCO = 30 - jdc;
      }

    //Ajustement de l heure en fonction de la periode ete hivers
      m = now.month();
      j = now.day();
      h = now.hour();

      if (( m < 3 || m > 10 ) || ( m == 3 && j < JCM ) || ( m == 10  && j > JCO ) || ( m == 3 && j == JCM && h <
2 ) || ( m == 10 && j == JCO && h > 2)) {
        //Serial.print ("Heure Hivers");
        RTC.adjust(epoch + 3600);
      }
      else{
        //Serial.print ("heure Ete");
        RTC.adjust(epoch + 7200);
      }

      //Serial.println("RTC Configured:");
    }
    else{
      //Si pb connexion au net, reset de l arduino. Necessaire en cas de
      //coupure de courant lorsque la box met du temps à revenir
      //Serial.print("Fatal Error: Unable to obtain DHCP address");
      lcd.setCursor(0, 0);
      lcd.print("Pb Connexion           ");
      lcd.setCursor(0, 1);
      lcd.print("Internet                ");
      delay(2000);
      TraitReset();
    }

  //definition des pages internet
  Ethernet.begin(mac, ip);
  webserver.setDefaultCommand(&defaultCmd);
  webserver.addCommand("index.html", &defaultCmd);
  webserver.addCommand("private.html", &privateCmd);
  webserver.addCommand("bt1.html", &bt1Cmd);
  webserver.addCommand("bt2.html", &bt2Cmd);
  webserver.addCommand("bt3.html", &bt3Cmd);
  webserver.addCommand("res.html", &resCmd);
  webserver.addCommand("mem.html", &memCmd);
  webserver.addCommand("aff.html", &pageWeb);
```

```
  webserver.begin();
}

void loop()
{
  //Recuperation date et heure
  //Pour le passage heure hivers et ete passage a 2h et 3h
  DateTime now = RTC.now();
  if  (( now.hour() == 2 ) && ( now.minute() == 1 ) && ( now.second() == 0 )) TraitReset();
  if  (( now.hour() == 3 ) && ( now.minute() == 1 ) && ( now.second() == 0 )) TraitReset();
  //Bornage des temperatures de consigne
  if (Tc > 22) Tc=22;
  if (Tm > 22) Tm=22;
  if (Te > 19) Te=19;

    Ta = dht.readTemperature();
    Hu = dht.readHumidity();


  //--------------------------------------------------------------------
  char buff[64];
  int len = 64;

  // Test falcultatif :
  // Je profite du module pour surveiller si la porte de garage est ouverte
  // si capteur en position ouvert la porte est ouverte et
  // dans ce cas on fait clignoter l afficheur LCD+
  if ((PINC & (1<<1)) == 0) {
    if ((PINH & (1<<3)) != 0) {
         PORTH &= ~(1<<3);
    }
    else{
         PORTH |= 1<<3;
    }
  }

  // bouton 3 et 4 appuye en meme temps - Memorisation de la situation du thermostat
  if (((PINC & (1<<5)) != 0)&((PING & (1<<1)) != 0)) {

    Memory();
    delay(300);
  }
  else{
      // bouton 1 et 2 appuye en meme temps - Reset
    if (((PINA & (1<<6)) != 0)&((PINA & (1<<2)) != 0)) {
    TraitReset();
    }else{
  //Traitement selon le bouton qui a ete appuye
  // 1 - Changement de mode (Off, Manuel, Programme)
  // 2 - Decrementation temperature de consigne
  // 3 - Incrementation temperature de consigne
  // 4 - Allume LED afficheur LCD  (LED+)
  //
    if ((PINA & (1<<6)) != 0) TraitBt(1);
    if ((PINA & (1<<2)) != 0) TraitBt(2);
    if ((PINC & (1<<5)) != 0) TraitBt(3);
    if ((PING & (1<<1)) != 0) TraitBt(4);
  // Attente anti-rebond
    delay(250);
   }
  }

  if (ST1 == 0)
  {
     if (etatChaud == true) Arr_Chauff();
      PORTH |= 1<<4;
  }
  else{
    if (ST1 == 1)
    {
      PG = pgm_read_word_near(PS + (now.dayOfWeek()));
      if (now.minute() > 30) {
        PR = 1;
      }else{
        PR = 0;
      }
```

```
    if ( PG == 1 ) Cons = pgm_read_word_near(PR1 + ((now.hour()*2)+ PR));
    if ( PG == 2 ) Cons = pgm_read_word_near(PR2 + ((now.hour()*2)+ PR));
    if ( PG == 3 ) Cons = pgm_read_word_near(PR3 + ((now.hour()*2)+ PR));
    if ( PG == 4 ) Cons = pgm_read_word_near(PR4 + ((now.hour()*2)+ PR));
    if ( Cons == 0 ){
      TpCons = Te;
    }
    else{
      TpCons = Tc;
    }
  }
  else{
    TpCons = Tm;
  }

  // Gestion du thermostat selon les tps
  DIFF = Ta - TpCons;
  if (DIFF > 0.5)
  {
    TPCHAUFF = 0;
  }
  else{
    if (DIFF < -0.8)
    {
      TPCHAUFF = 540000;
    }
    else{
      TPCHAUFF = (5 - DIFF / 0.2) * 60000;

    }
  }

  TPARRET = 600000 - TPCHAUFF;
  ACTUELLE = millis();
  if (Mode == "") {
    Mode = "C";
    STARTC = millis();
    Act_Chauff();
  }
  if (Mode == "A") {
    if (TPCHAUFF != 0) {
      if(ACTUELLE - STARTA > TPARRET) {
        Mode = "C";
        STARTC = ACTUELLE;
        Act_Chauff();

      }
    }
  }
  if (Mode == "C") {
    if(ACTUELLE - STARTC > TPCHAUFF) {
      Mode = "A";
      STARTA = ACTUELLE;
      Arr_Chauff();
    }
  }

}



//Mise a jour afficheur LCD 2x16
//Premiere ligne
lcd.setCursor(0, 0);
  if (ST1==0) {
  lcd.print(" Off    ");
  //digitalWrite(PIN_LED_1, LOW);
  PORTA &= ~(1<<4);
  }
  else{
  //digitalWrite(PIN_LED_1, HIGH);
  PORTA |= 1<<4;
  }
   if (ST1==1) {
  if (Cons==0){
```

```
      lcd.write((uint8_t)0);
    }else{
      lcd.write((uint8_t)1);
    }

                lcd.print(TpCons, 1);
    lcd.print(" ");
                if ( PG == 1 )lcd.write((uint8_t)2);
                if ( PG == 2 )lcd.write((uint8_t)3);
                if ( PG == 3 )lcd.write((uint8_t)4);
                if ( PG == 4 )lcd.write((uint8_t)5);
                lcd.print(" ");
    }
    if (ST1==2) {
    lcd.print("Man");
    lcd.print(TpCons, 1);
    lcd.print(" ");
    }
    lcd.print(wd[now.dayOfWeek()]);
    lcd.print(" ");
    lcd.print(now.day());
    lcd.print('/');
    lcd.print(now.month());
          lcd.print("  ");

    //Second ligne
    lcd.setCursor(0, 1);
    if (etatChaud == true){
    lcd.print((char)127);
    }else{
    lcd.print(" ");
    }
    lcd.print(Ta, 1);
    lcd.print((char)178);
    lcd.print(" ");
    lcd.print(Hu, 0);
    lcd.print((char)37);
    lcd.print(" ");
    lcd.print(format2(str, (now.hour())));
    lcd.print(":");
    lcd.print(format2(str, (now.minute())));
          lcd.print(":");
    lcd.print(format2(str, (now.second())));
    lcd.print(" ");


  // Traitement des connexions http
  webserver.processConnection(buff, &len);
}

//Lancement de la chaudiere
void Act_Chauff() {
  etatChaud=true;
  //Serial.print(F("\nActivation Chaudiere"));
  //Serial.print(TPCHAUFF);
      PORTA |= 1<<0; //LED2 HIGH
      PORTL |= 1<<5; //Relai HIGH
}

//Arret de la chaudiere
void Arr_Chauff() {
  etatChaud=false;
  //Serial.print(F("\nArret Chaudiere"));
  //Serial.print(TPARRET);
  PORTA &= ~(1<<0); //LED2 LOW
  PORTL &= ~(1<<5); //Relai LOW
}

//Traitement selon le bouton qui a ete appuye
// 1 - Changement de mode (Off, Manuel, Programme)
// 2 - Decrementation temperature de consigne
// 3 - Incrementation temperature de consigne
// 4 - Allume LED afficheur LCD  (LED+)
//
void TraitBt (byte PressBt) {
  switch (PressBt) { // debut de la structure
```

```
  case 1:
    Mode = "";
    ST1++;
    if ( ST1 > 2 ) ST1=0;
    break;
  case 2:
    Mode = "";
    if (ST1 == 1) {
      if ( Cons == 1)Tc += t;
      if ( Cons == 0)Te += t;
    }
    if (ST1 == 2) Tm += t;
    if (Tc > 22) Tc=22;
    if (Te > 19) Te=19;
    if (Tm > 22) Tm=22;
    break;
  case 3:
    Mode = "";
    if (ST1 == 1) {
      if ( Cons == 1)Tc -= t;
      if ( Cons == 0)Te -= t;
    }
    if (ST1 == 2) Tm -= t;
    if (Tc < 16) Tc=16;
    if (Te < 16) Te=16;
    if (Tm < 16) Tm=16;
    break;
  case 4:

    if ((PINH & (1<<3)) != 0) {
         PORTH &= ~(1<<3);
    }
    else{
         PORTH |= 1<<3;
    }
    break;
  }
}



void Memory() {
  //Serial.print(F(" Memory OK..."));
  delay(1000);
  lcd.print("Memory...");
  delay(1000);
  EEPROM.write(0,Tc);
  EEPROM.write(1,Te);
  EEPROM.write(2,Tm);
  EEPROM.write(3,ST1);
  lcd.clear();
  lcd.print("Memory OK");
  delay(1000);
}



char* format2(char* str, byte value) {
  sprintf(str, "%02d", value);
  return str;
}



unsigned long getNTP() {
  sendNTPpacket(timeServer); // send an NTP packet to a time server

    // wait to see if a reply is available
  delay(1000);
  if ( Udp.parsePacket() ) {
    // We've received a packet, read the data from it
    Udp.read(packetBuffer,NTP_PACKET_SIZE); // read the packet into the buffer

    //the timestamp starts at byte 40 of the received packet and is four bytes,
    // or two words, long. First, esxtract the two words:
```

```
    unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
    unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
    // combine the four bytes (two words) into a long integer
    // this is NTP time (seconds since Jan 1 1900):
    unsigned long secsSince1900 = highWord << 16 | lowWord;
    //Serial.print("Seconds since Jan 1 1900 = " );
    //Serial.println(secsSince1900);

    // now convert NTP time into everyday time:
    //Serial.print("Unix time = ");
    // Unix time starts on Jan 1 1970. In seconds, that's 2208988800:
    const unsigned long seventyYears = 2208988800UL;
    // subtract seventy years:
    unsigned long epoch = secsSince1900 - seventyYears;

    return epoch;
  }
}

// send an NTP request to the time server at the given address
unsigned long sendNTPpacket(IPAddress& address)
{
  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;

  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer,NTP_PACKET_SIZE);
  Udp.endPacket();
}


void JDS () {
  if ( m1 < 3 ) {
    z = y1-1;
    b = 0;
  }
  else{
    z = y1;
    b = 2;
  }
  d1 = (((((23*m1)/9) + 31 + 4 + y1 + (z/4)) - (z/100) + (z/400)) - b);
  jdc = d1 %7;
}

void TraitReset() {
  lcd.clear();
  lcd.print("Reset ...");
  delay(500);
  PORTH &= ~(1<<4);; //Pulling the RESET pin LOW triggers the reset.
```