MISE EN SITUATION

Le thermostat contrôle 60 % des factures énergétiques. Le principe de la commande de chauffage est assez simple dans son concept : on fixe une température de confort, et le système de mesure active ou non le chauffage en fonction de la température ambiante mesurée.

Ce qui signifie en autre qu'un tel thermostat est capable de :

- Mesurer la température
- Définir la plage de commande de la chaudière
- Transmettre l'ordre de commande

Problématique :

Comment automatiser la gestion programmable du chauffage à partir d'un seul capteur numérique ?

Pour réponde à cette problématique il nous faut :

- Récupérer les valeurs de température mesurées par le capteur électronique,
- Afficher sur écran LCD la température en temps réel
- Commander le chauffage automatiquement en fonction des seuils fixés pour la régulation.

Ce dernier exercice a pour objectif de vous apprendre à programmer sur Arduino le circuit DS1621 étudié auparavant. Vous disposez pour cela :

- Du document ressource détaillant le cahier des charges et expliquant le principe de la programmation I2C sous Arduino
- Du fichier de simulation « Simulation.DSN » contenu dans le dossier ressources
- Du programme Arduino à compléter situé dans le dossier ressource

Cahier des charges :

On désire une température de confort de 21°C avec une hystérésis de +/- 1,5°C. Le thermostat doit donc être réglé avec :

TL = 19,5°C et TH = 22,5°C

La conversion doit être continue.

La sortie du thermostat est fixée à 1 au repos. La commande de chauffage se fera donc sur un niveau logique 0.

PROGRAMMATION DE LA COMMUNICATION I²C AVEC ARDUINO

La bibliothèque **Wire** permet de communiquer sur le bus I²C en Arduino. Il est nécessaire de l'importer dans votre programme :

```
#include <Wire.h>
```

Description des fonctions I2C :

Wire.**begin**() : initialise le mode de communication I2C. À écrire une fois dans la procédure **setup**( )

Wire.**beginTransmission**(**adress**) : débute une transmission avec un circuit I2C. Cette fonction permet de prendre la main sur le bus, et d'indiquer, par la variable **adress,** avec quel circuit l'on désire communiquer.

Wire.**requestFrom**(***adress***, ***quantity***, ***STOP***) :

Requête envoyée à l'esclave (***adress***) lui demandant de placer sur le bus I2C les données de son ou de ses registres. Le bit R/W est alors automatiquement mis à 1 (Lecture).

Quantity correspond au nombre **d'octets** à lire. ***STOP*** est un booléen (True/False) indiquant si l'on doit placer un stop après exécution de la requête pour libérer le bus.

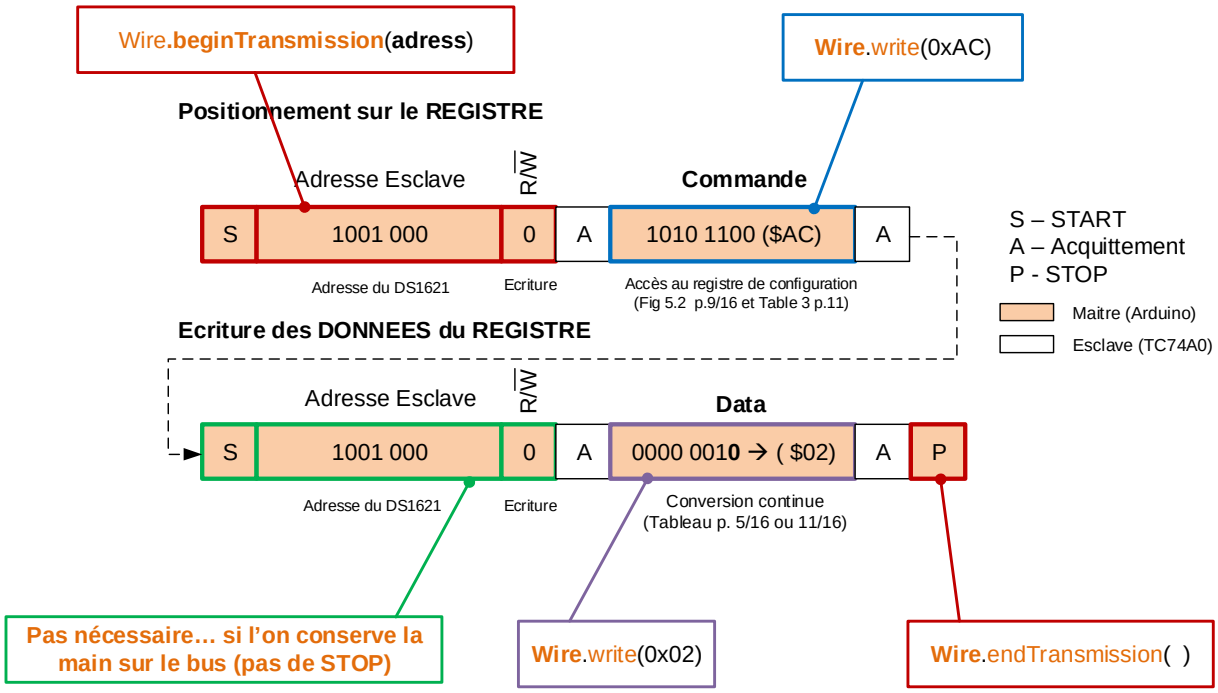
Wire.**available**( ) : vérifie si les données sont disponibles sur le BUS après le lancement d'une requête. Cette fonction renvoie le nombre d'octets disponibles.

Wire.**read**( ) : Permet de lire un ou plusieurs octets placés par l'esclave sur le bus. Cette fonction ne peut fonctionner que si un début de transmission est effectué et si les données sont disponibles.

Wire.**endTransmission**( ) : Termine une transmission en cours sur le bus I2C. Cette fonction permet de libérer le bus en plaçant un STOP sur la trame.

Wire.**write**(*byte*) : Permet d'écrire un octet byte sur le bus. Cette fonction ne peut fonctionner que si un début de transmission est effectué. Le bit R/W est alors automatiquement mis à 0.

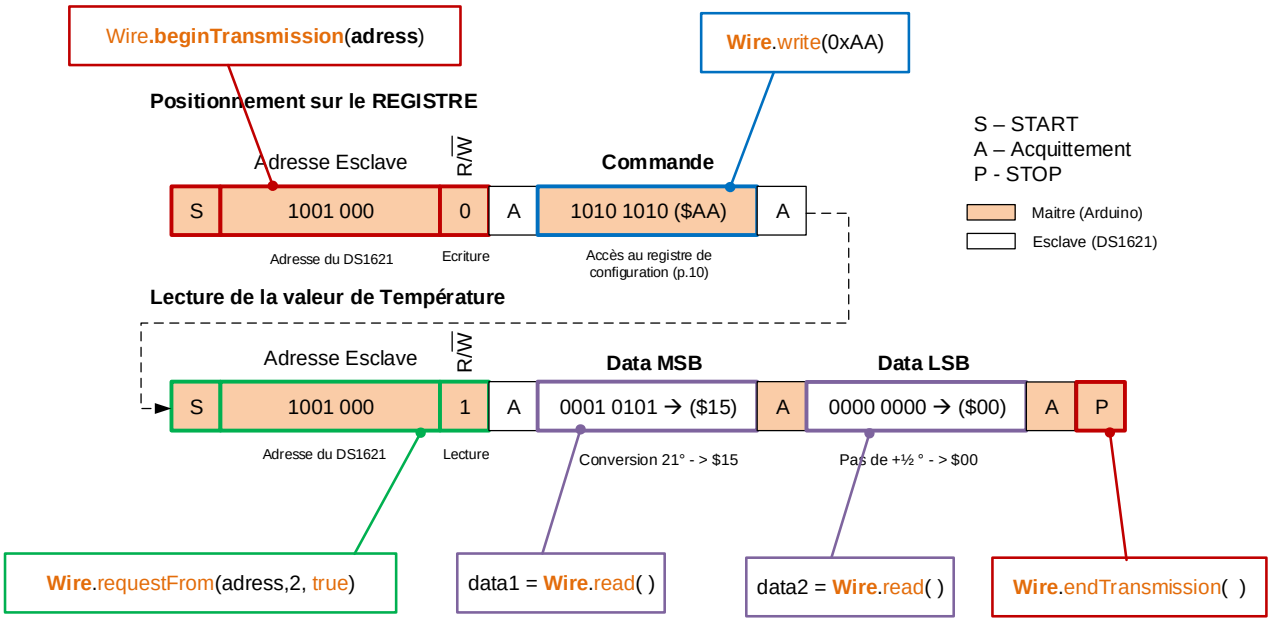
Exemple de Programme en Écriture :



Ce qui se traduit en Arduino par :

```
Wire.begin();                          // Initialisation du module I2C
Wire.beginTransmission(DEV_ID);        // START - Prise en main du BUS et connexion au circuit DS1621
delay(10);
Wire.write(0xAC);                      // Acces au registre Config
Wire.write(0x02);                      // Mise en Conversion continue
delay(10);
Wire.endTransmission();                //STOP - Libération du BUS
```

Exemple de programme en Lecture :



Ce qui se traduit en Arduino par :

```
Wire.beginTransmission(DEV_ID);              // START - Prise en main du BUS et connexion au circuit DS1621
Wire.write(0xAA);                            // Commande de lecture de température
Wire.requestFrom(DEV_ID, 2, true);           //Lecture de 2 octets
if (Wire.available()) {                      //Si les données sont disponibles...
  firstByte = Wire.read();                   //lecture 1ere Octet
  secondByte = Wire.read();                  //lecture 2eme Octet (1/2 degré)
}
delay(10);
Wire.endTransmission();                       //STOP - Libération du BUS
```

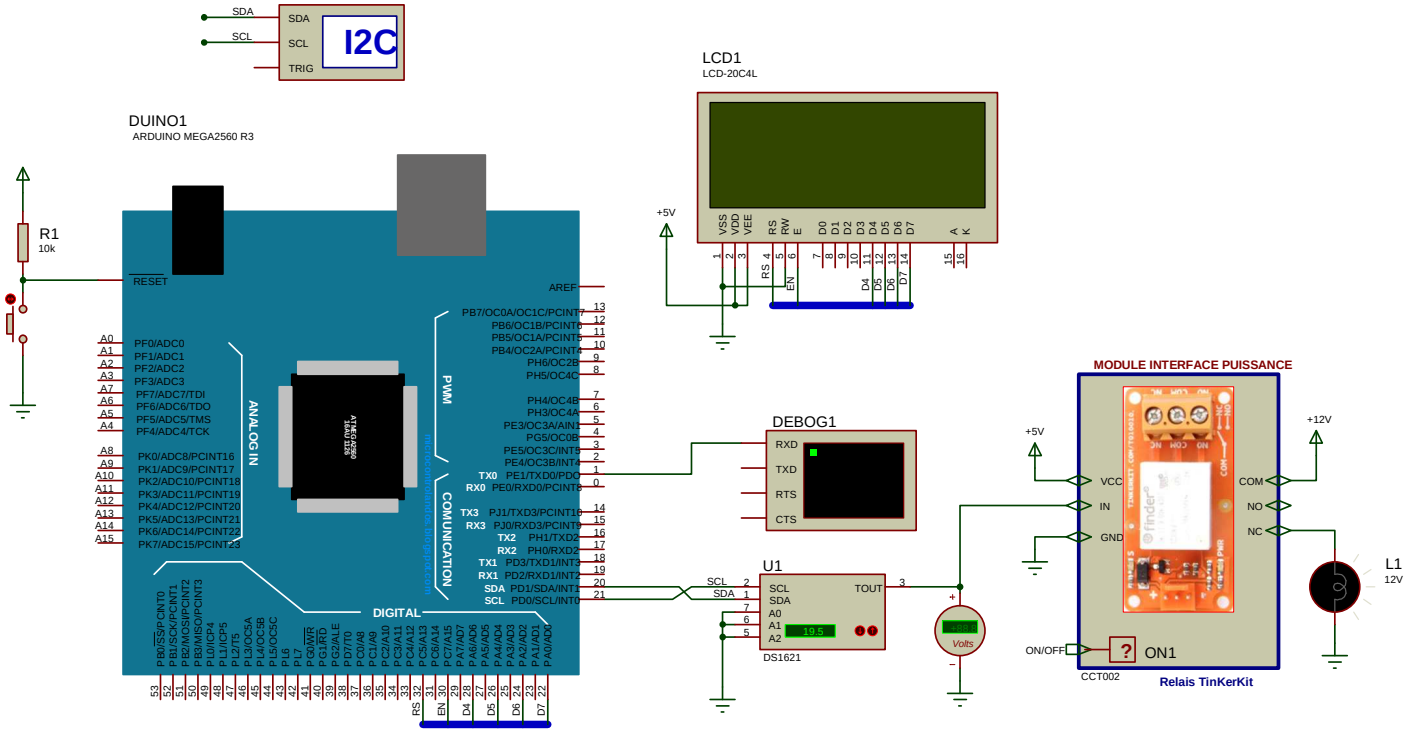
TRAVAIL DEMANDÉ

♦ RÉALISATION du PROGRAMME :

☞ Complétez les différentes parties manquantes du programme Arduino en vous aidant du cahier des charges et des documents d'aide mis à votre disposition.

♦ SIMULATION et MISE AU POINT du PROGRAMME :

☞ Ouvrez le logiciel de simulation **Proteus ISIS** puis chargez le fichier de simulation **Simulation.DSN**



☞ Placez le programme Arduino « *ArduinoDS1621.ino.hex* » dans le microcontrôleur et simulez le montage. Mettre au point le programme afin de répondre au cahier des charges.