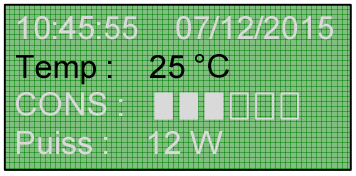
♦ PRESENTATION DU PROJET :

On vous demande d'étudier le concept de gestion de ventilation automatisée des boitiers de poste informatique.

On désire **Mesurer** la température interne du boîtier **toute les 5 secondes**. Pour cela, il vous faudra mettre en place l'acquisition et la conversion de la température à partir d'un capteur connecté sur bus I2C.

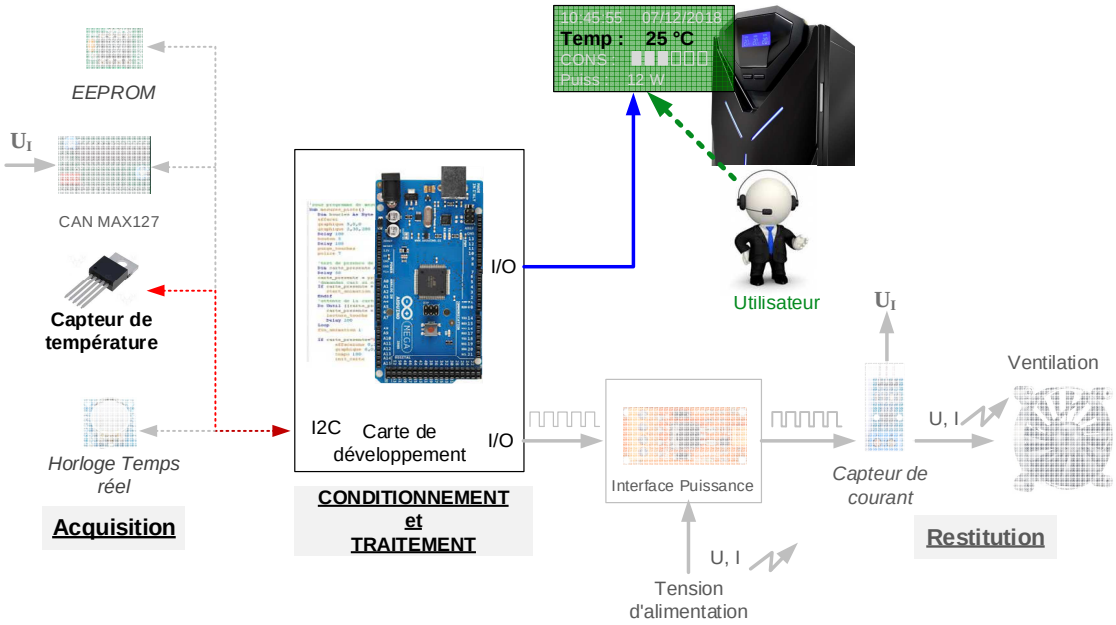
Il nous faut dans un premier temps afficher ces informations en LOCAL :



Se pose alors plusieurs problématiques auxquelles nous allons répondre aux travers des différents TP mettant en œuvre le principe du bus I2C :

- ***comment mesurer la température à l'intérieur du boitier ?***
- *comment connaitre l'heure et la date indépendamment du PC ?*
- *comment gérer la vitesse d'extraction de chaleur du boitier en fonction de la température ?*
- *comment mesurer la puissance électrique consommée par le ou les ventilateurs ?*

Nous nous attacherons dans cette partie à répondre au besoin de mesure de la température:



Problématique posée : *Comment mettre en œuvre la mesure en temps réel de la température sur un système microprogrammé ?*

Le capteur de température utilisé est le TC74A, capteur numérique communicant sur BUS I2C.

Cette Activité est évaluée par revue de projet individuelle !

♦ Programmation de la communication avec le circuit TC74A0 en 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. A 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** ( ) : Permet d'écrire une donnée 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 :

```
Wire.beginTransmission(ADRESS);        //Début de la transmission
Wire.write(0x04);                      //Commande  : modification de la CONFIGURATION
Wire.write(0x10);                      //écriture de $10. : Passage en mode Standby
Wire.endTransmission( );               // Arrêt de transmission : libération du BUS
```

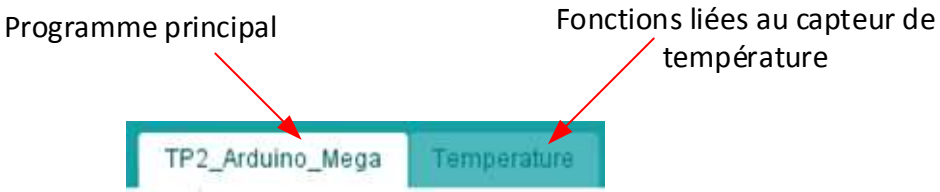
Exemple de programme en Lecture :

```
Wire.beginTransmission(ADRESS);        // Début de transaction I2C
Wire.write(0x00);                      //Commande : lecture de la donnée
Wire.requestFrom(ADRESS, 1, true);     // requête de lecture d'un octet avec STOP
if ( Wire.available( ) ) {             // Si la donnée est présente sur le BUS
        data = Wire.read( );           // Lecture de la donnée
} else {
        bError = True;                 // Sinon erreur
}

delay(10);

Wire.endTransmission( );               // Arrêt de transmission : libération du BUS
```

♦ **REALISATION du PROGRAMME** :

☞ Ouvrez le programme « *TP2_Arduino_Mega.ino* » contenu dans le dossier de ressources « *programme* »

Celui-ci est organisé en Onglets :

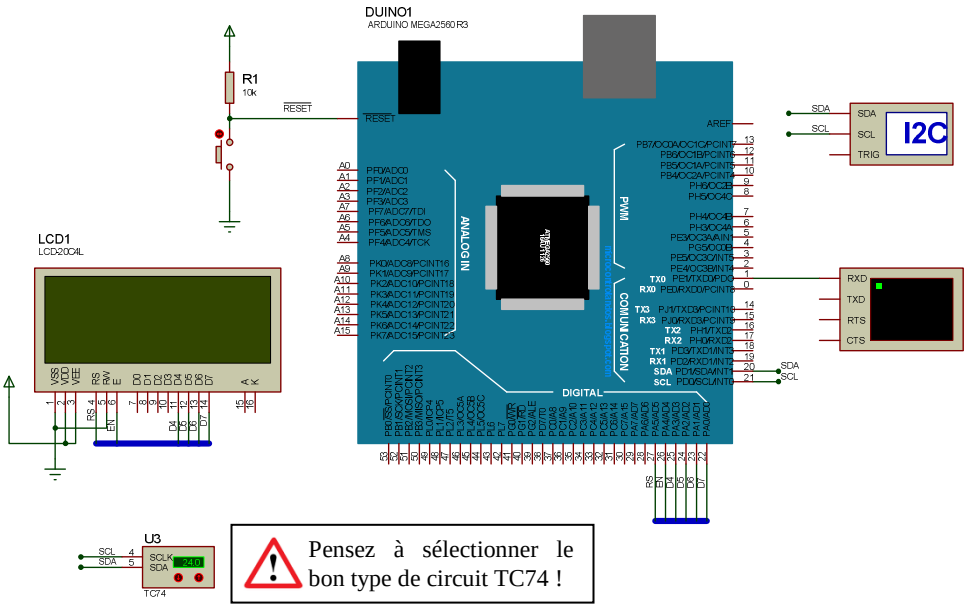


☞ Complétez les différentes parties du programme en vous aidant du cahier des charges, à savoir :

- Procédure **AffichageTemp**( )
  Cette procédure permet d'afficher sur l'écran LCD et d'envoyer en série pour le débogage la température mesurée par le capteur en °C (utilisez (char)**223** pour le symbole °).

- Fonction *byte* **getTempTC74**( )
  Cette procédure permet de lire la température sur le circuit TC74A. La fonction renvoie un octet contenant la valeur numérique de température.

♦ **SIMULATION et MISE AU POINT du PROGRAMME** :



☞ Ouvrez le fichier de simulation **Proteus ISIS** « *TP2-TEMP-I2C.DSN* » contenu dans le dossier de ressources « \simulation »

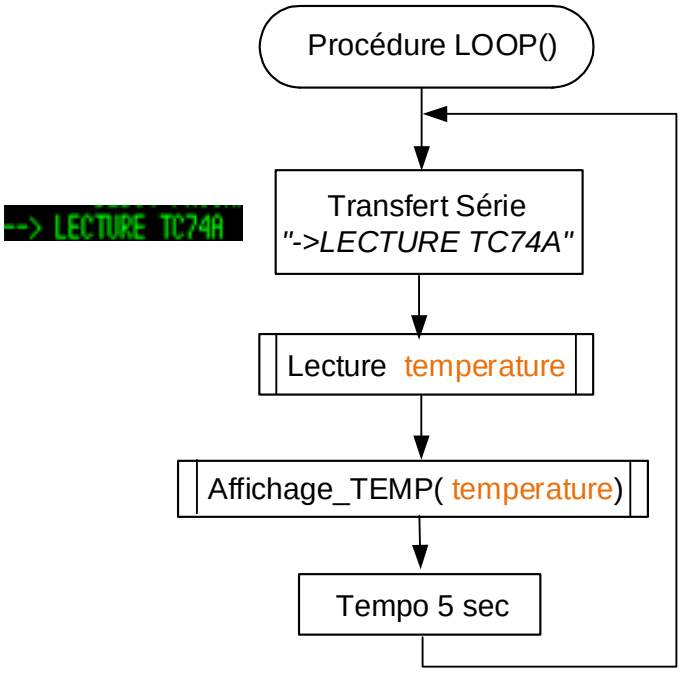
☞ Placez le programme ***TP2_Arduino_Mega.ino.hex*** dans le modèle de simulation Arduino, et simulez le montage. Mettre au point le programme afin de répondre au cahier des charges.

TRAVAILLEZ PAR ÉTAPE !!! Utilisez les opérateurs /* et */ pour mettre en commentaire les fonctions ou parties de programme en attentes d'écriture.

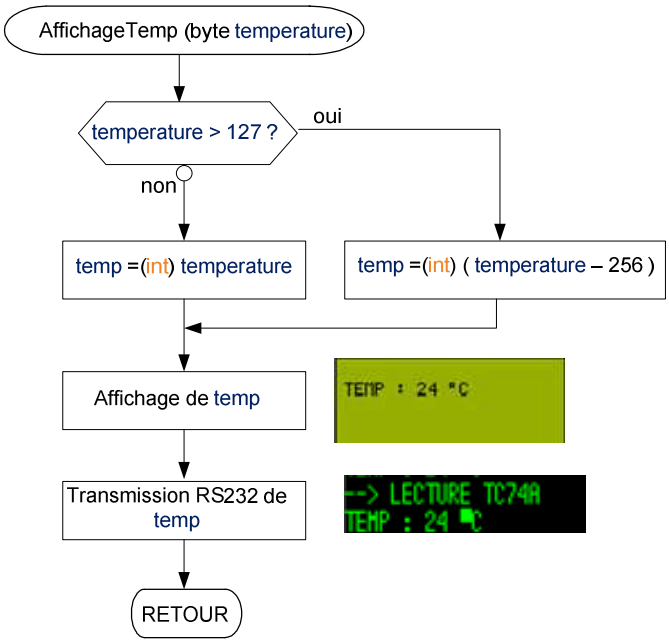
Utilisez le débogueur I2C pour analyser le contenu des trames !

♦ **ORGANIGRAMMES DES FONCTIONS :**

Programme PRINCIPAL (procédure LOOP) :



SP AffichageTemp(*byte* temperature) :



SP Initialisation () :