ACTIVITÉ 1 - Communication par BUS I²C

Étude du capteur de température-thermostat DS1621

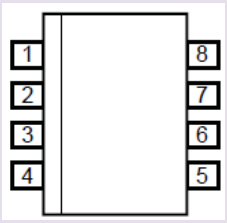
ETAPE 1 : Étude du capteur de température DS1621

Q1 : A partir de la documentation technique du circuit **DS1621**, complétez les éléments suivants :

/2

- Tension **d'alimentation** du circuit : .....
- Gamme de température **mesurée** : .....
- Temps de conversion maximal de la Température : .....
- **Précision** de mesure de conversion Température : .....
- Complétez le tableau du **brochage** du circuit format DIP (en Français !!) :

/3



Broche	Nom	Rôle/valeur
1		
2		
3		
4		
5 à 7		
8		

Q2 : A partir de la Table 2 du DT (P. 4/16) de la documentation technique du circuit DS1621, complétez les éléments suivants :

- Complétez le contenu du **registre de température** du circuit pour les informations suivantes

Température	Octet de poids FORT (MSB)	Octet de poids FAIBLE (LSB)	HEXA
0°C			
-10 °C			
+ 26°C			
+ 16,7 °C			
+130 °C			
- 60°C			
+ 25,5 °C			

/7

- **Complétez les adresse esclaves des différents circuits DS1621 suivant en <u>écriture</u> (DT pages 8 et 9) :**

/7

Réf. circuit	D7	D6	D5	D4	**A2**	**A1**	**A0**	R/$\overline{W}$	Adresse HEXA
Adresse0					**0**	**0**	**0**		
Adresse1					**0**	**0**	**1**		
Adresse2					**0**	**1**	**0**		
Adresse3					**0**	**1**	**1**		
Adresse4					**1**	**0**	**0**		
Adresse5					**1**	**0**	**1**		
Adresse6					**1**	**1**	**0**		

Q3 : Démarrage de la conversion de Température :

- Complétez le contenu de la trame suivante si l'on désire "activer" la conversion de température du capteur (**DT page 10**) :

  On prendra dans tout l'exemple A0 = A1 = A2 = 0

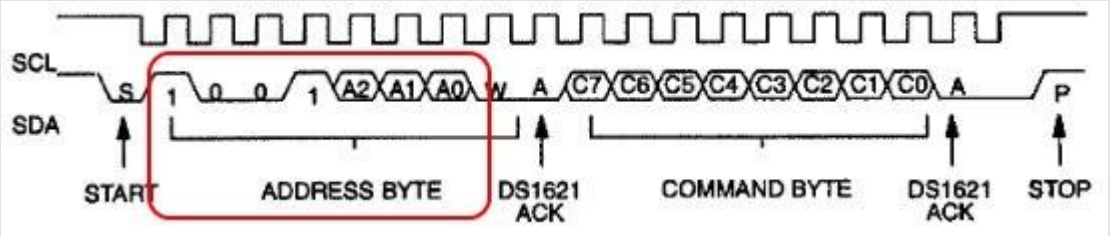
Ecriture dans le REGISTRE de Configuration

	Adresse Esclave	R/$\overline{W}$		**Commande**		
S	1001 000		A		A	P

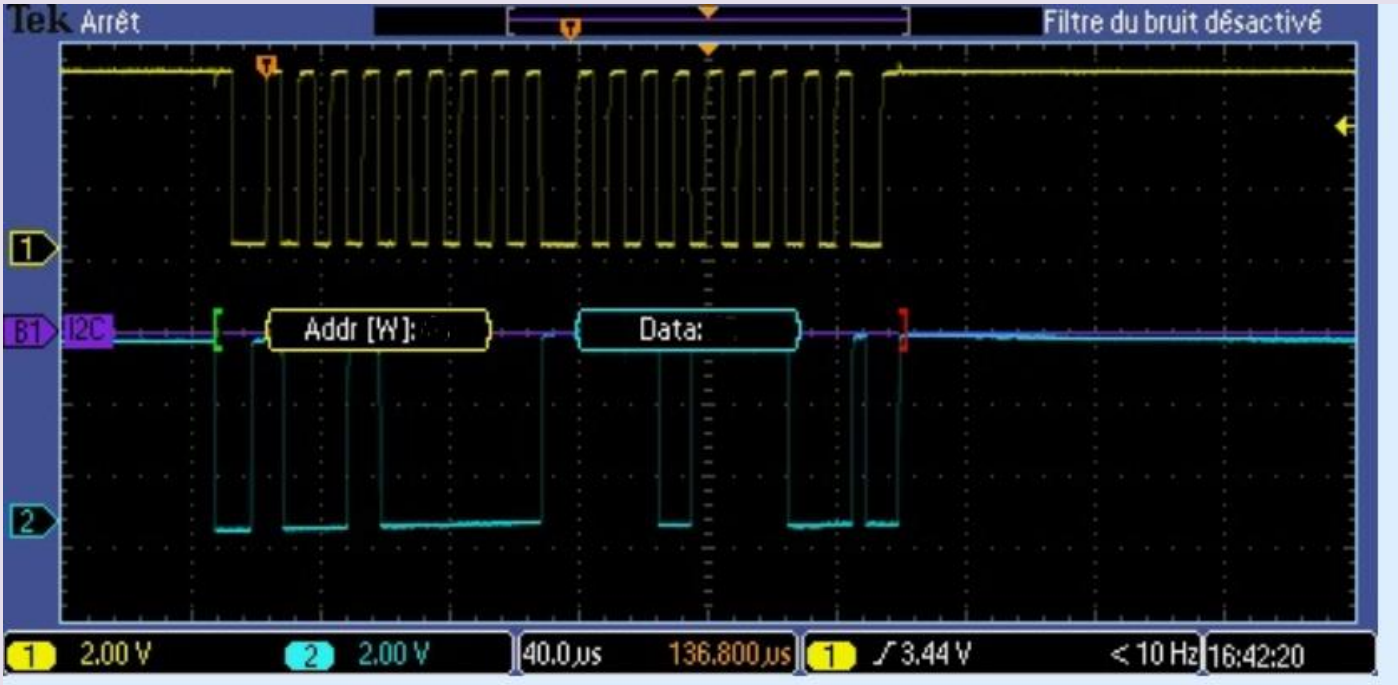
S – START
A – Acquittement
P - STOP

Maitre (Arduino)
Esclave (DS1621)

On rappelle que l'adresse d'un composant I^2C est codée avec sept bits (A0 à A6). Les bits de poids faible A0, A1 et A2 peuvent être configurés matériellement, par exemple en reliant les trois broches A0, A1 et A2 à la référence électrique GND comme sur le schéma du montage ci-dessus, on les met au niveau logique bas, A0 = A1 = A2 =0. Les bits A3 à A6 sont fixés par le fabricant du DS1621, ici A3=1, A4=0, A5=0 et A6=1.



Retrouver sur la trame ci-dessous l'adresse est la commande « activer » conversion.



Lecture de la température :

- Complétez le contenu de la trame suivante si l'on désire "activer" la conversion de température du capteur en supposant T = 21,3 °C (**DT page 10**) :

/6

Positionnement sur le REGISTRE

	Adresse Esclave	R/$\overline{W}$		**Commande**	
S	1001 000		A		A

Lecture de la valeur de Température

	Adresse Esclave	R/$\overline{W}$		**Data MSB**		**Data LSB**		
S			A		A		A	P

S – START
A – Acquittement
P - STOP

Maitre (Arduino)
Esclave (DS1621)

ETAPE 2 : PROGRAMMATION DE LA COMMUNICATION I2C AVEC ARDUINO

La bibliothèque **Wire** permet de communiquer sur le bus I^2C 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.

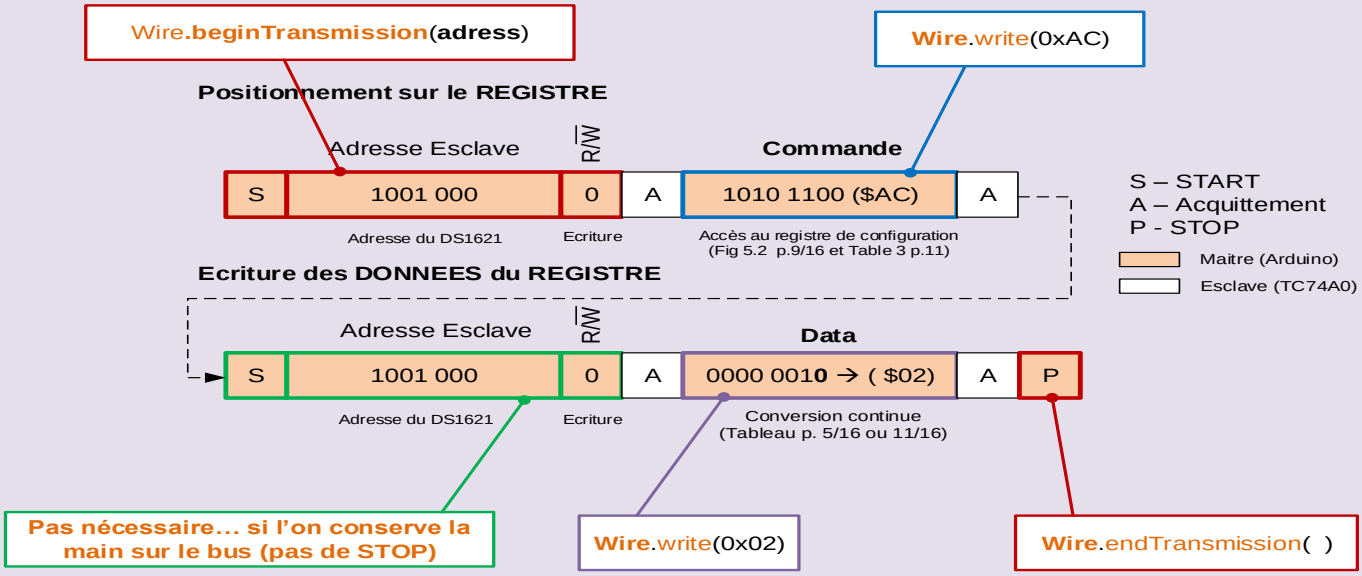
Wire.**available**( ) : vérifie si les données sont disponibles sur le BUS après le lancement d'une requête. Cette fonction renvoie le nombre d'octets disponibles.

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.

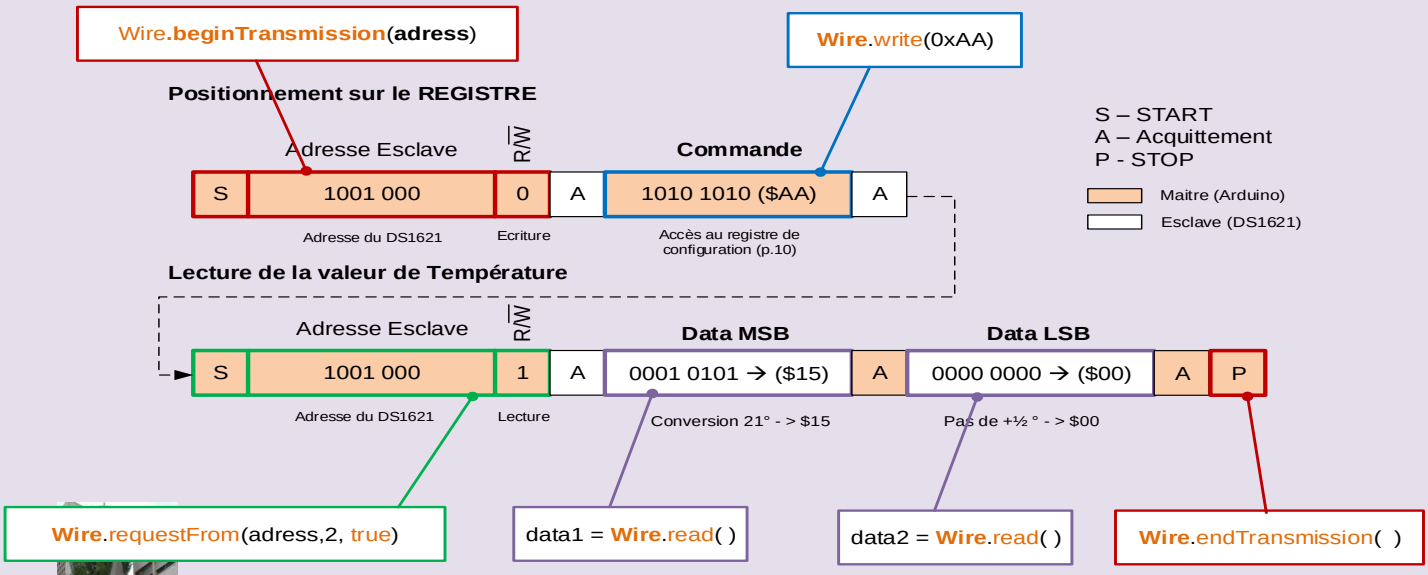
Exemple de Programme en Écriture :



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.

#include <Wire.h>

#define ADDRESS_DS1621 (……..)

/* Commandes du DS1621 */

#define READ_TEMPERATURE ……

#define ACCESS_CONFIG ……..

#define START_CONVERT ……..

#define STOP_CONVERT …….

void loop() {

/* Lancement de la conversion */

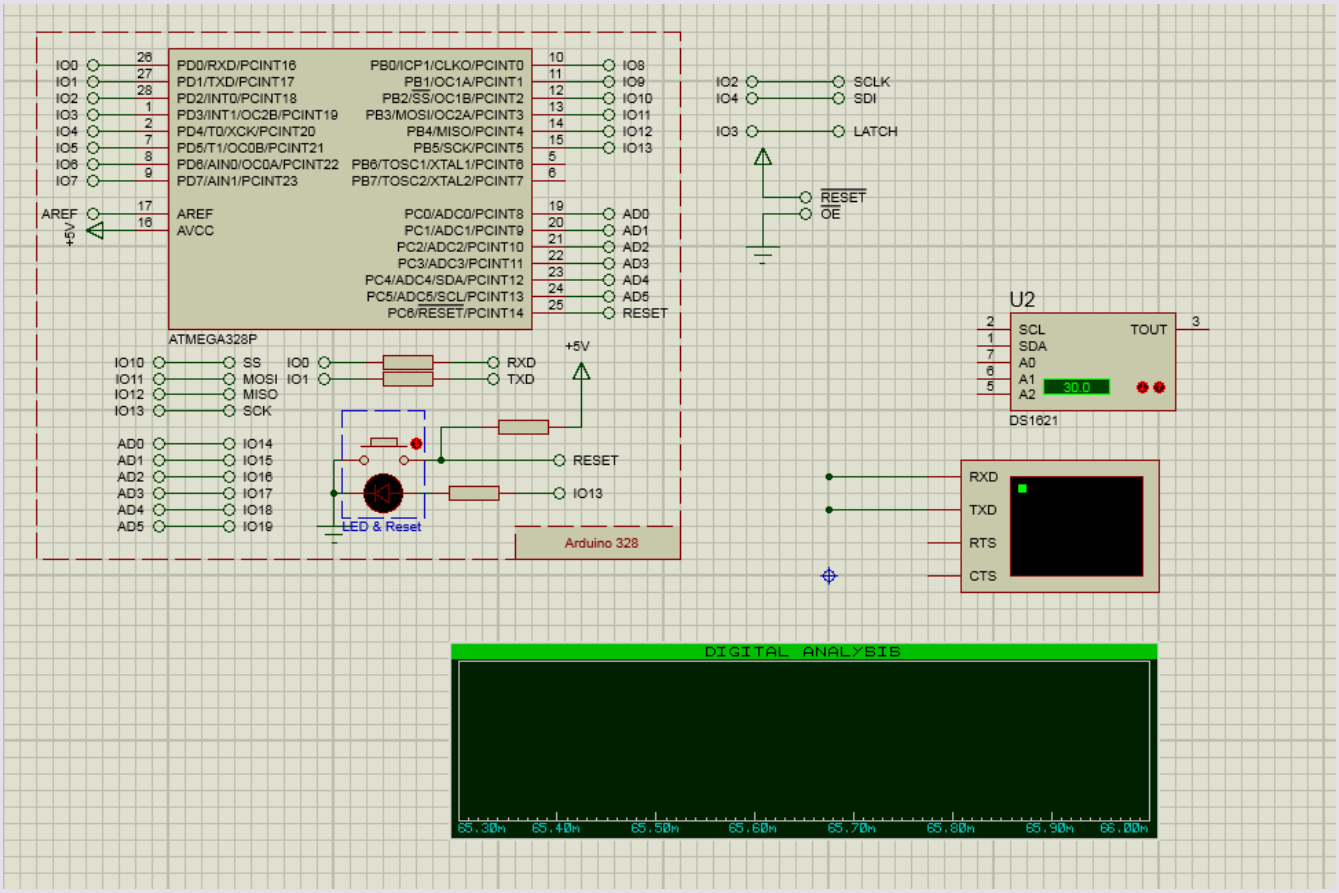
…….;// début prise en main du bus I2C et connection au DS1621

……..;// début conversion

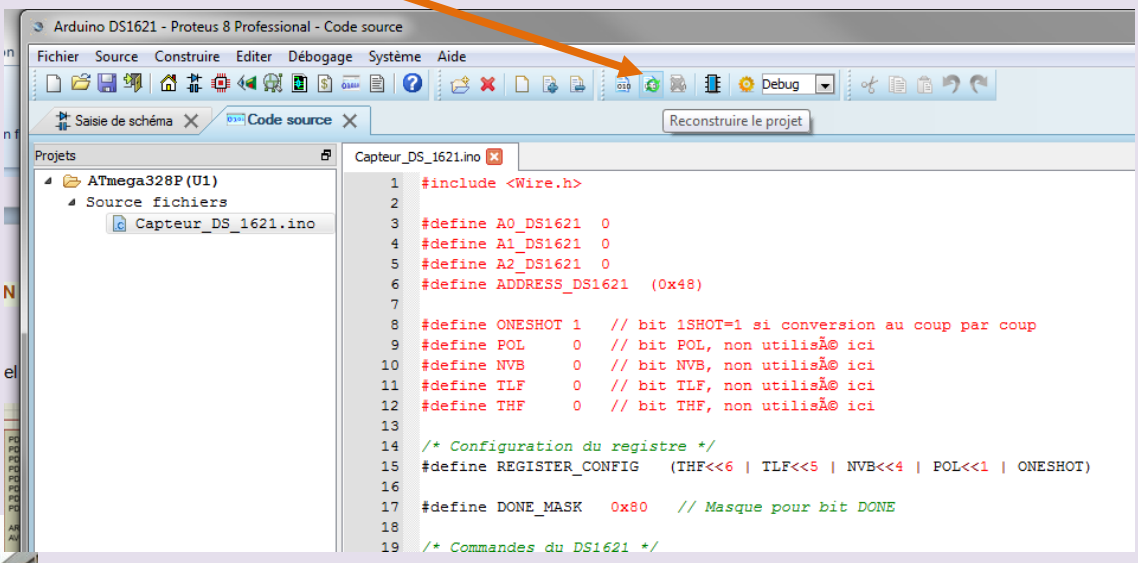
Wire.endTransmission(); // ……

ETAPE 3 : SIMULATION et MISE AU POINT du PROGRAMME :

☞ Ouvrez le logiciel de simulation **Proteus ISIS** puis chargez le fichier de simulation **Arduino Ds1661.DSN**

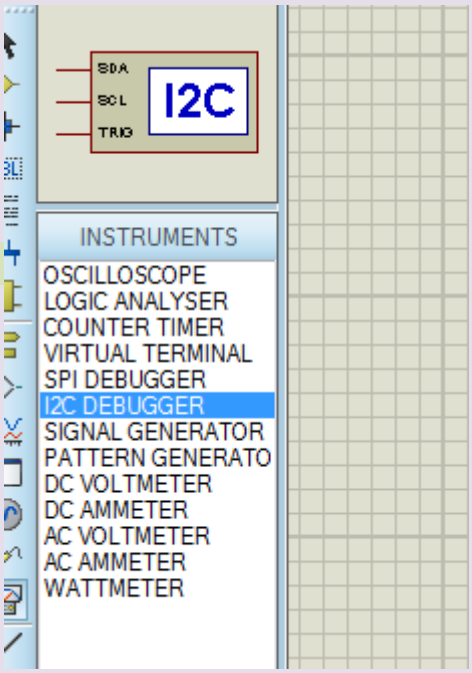


- ✓ **Compléter le schéma ci-dessous, connecter la carte arduino au capteur DS1621.**
- ✓ **Dans le Digital Analysis faire glisser le Bus I2C « SCL et SDA »**
- ✓ **Reconstruire le projet**



Si compilateur Avr arduino n'est pas installé, Clic sur ATMEGA328P , Editer propriétés et dans Program File insérer le programme « Capteur_DS_1621.ino.standard.hex »

- ✓ **Appuyer sur barre espace et commenter.**
- ✓ **Lancer la simulation et commenter.**



☛ Utilisation de l'outil I2C DEBUGGER

- ✓ Connecter L'I2C DEBUGGER et lancer la simulation.
- ✓ Retrouver sur L'I2C DEBUGGER les trames suivantes : Impression écran.

- **Démarrage de la conversion de Température :**
- **Lecture de la température :**