Nom Prénom

« COMMENT MESURER LA TEMPERATURE INTERIEURE D'UNE SERRE ? »

Semaine n°2 : SIN

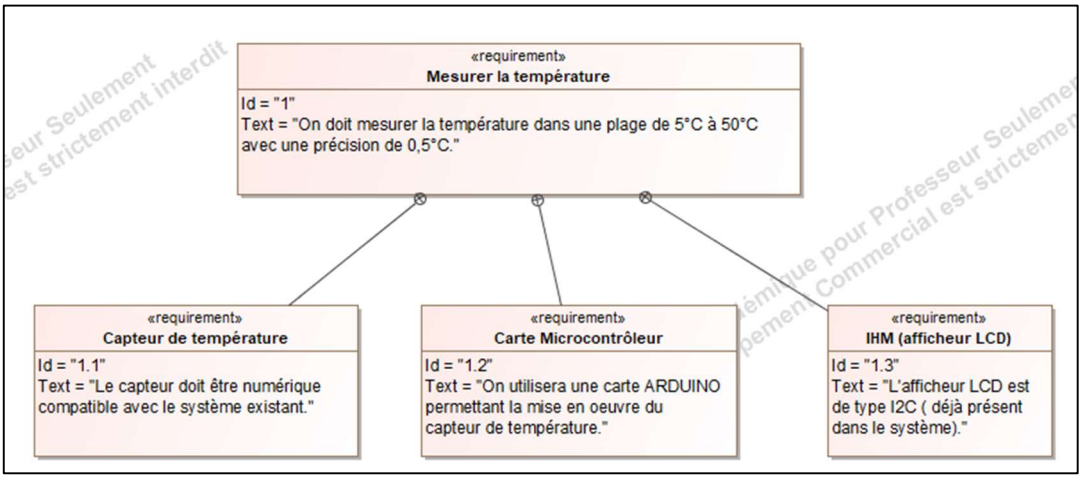
Présentation du cahier des charges

On souhaite mesurer la température intérieure d'une serre, avec une plage de mesure de 5°C à 50°C avec une précision de 0,5°C.

Le capteur choisi doit être :

-numérique
-très facilement interfaçable avec l'afficheur déjà présent sur le système

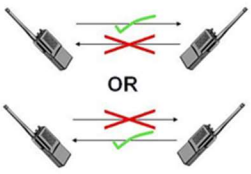
Le diagramme des exigences ci-dessous montre l'organisation de ce mini projet :



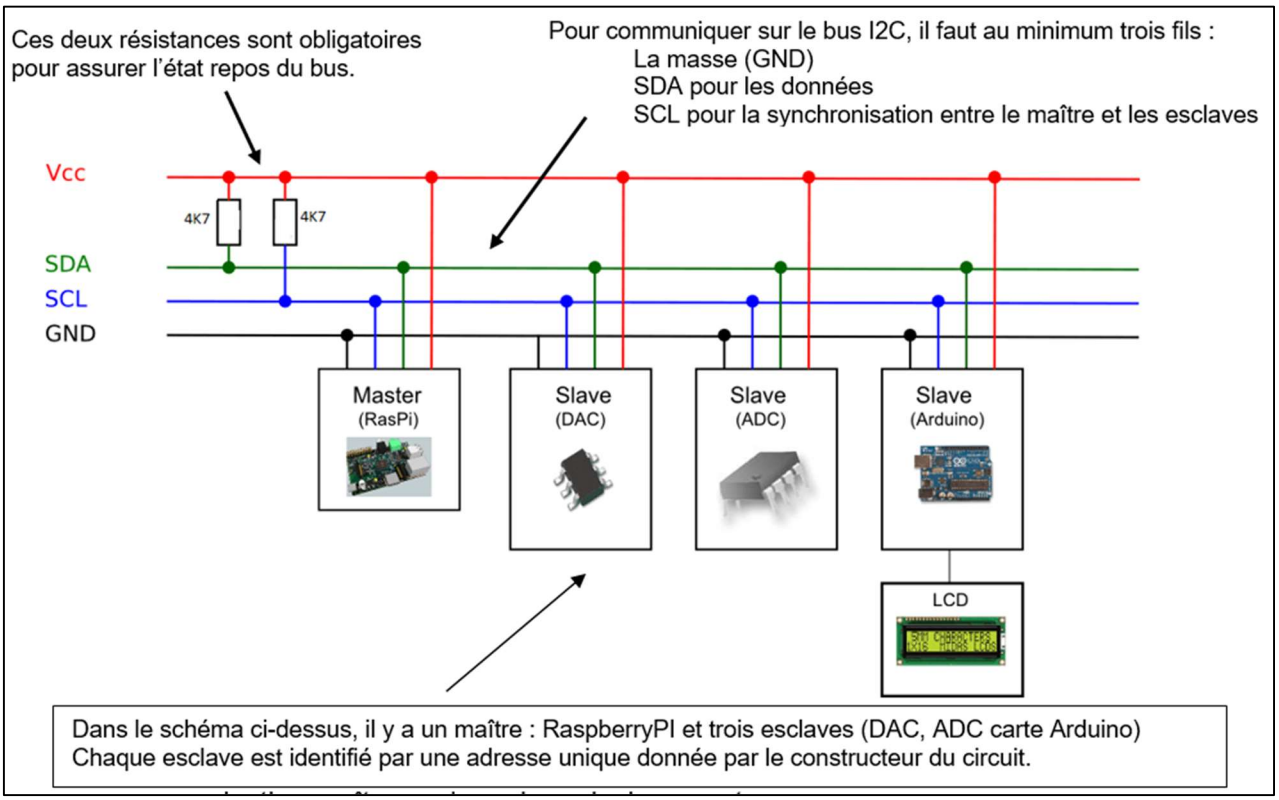
Q1/-Ouvrir le fichier ci-dessus et, déterminer le capteur correspondant au mieux à la description ci-dessus.
Votre choix doit être argumenté (technologie de sortie, protocole, précision, gamme de mesure, interfaçage avec le système existant).

Comment communiquer avec le bus I2C ?

Le bus I2C est un bus de série synchrone bidirectionnel half duplex.
La communication est de type maître esclave donc pas de perte de données.
Le débit de la communication est :
100K bits/s : mode standard
5M bits/s : mode ultra fast



Présentation de l'organisation matérielle du bus I2C ou (TWI : Two Wire Interface) et quelques principes de bases.



Le bus I2C possède deux lignes importantes :

SDA (Serial Data Line) pour les données
SCL (Serial Clock Line) pour la synchronisation des données entre le maître et les esclaves.

Le protocole I2C utilise le principe : Maître/esclave :

un seul maître : Il gère tous les échanges sur le bus. (remarque très importante : le maître ne possède pas d'adresse I2C)
chaque esclave possède une adresse codée sur 7 bits pour que le maître puisse communiquer.

Q2/ -Dans le schéma ci-dessus :

-donner le nom du composant faisant office de maître du bus.

-le nombre d'esclaves et le type de circuit.

Q3/-Combien d'esclaves peut-on câbler sur un bus I2C ?
Justifier votre réponse.

Analyse de la trame I2C : forme des signaux SDA et SCL

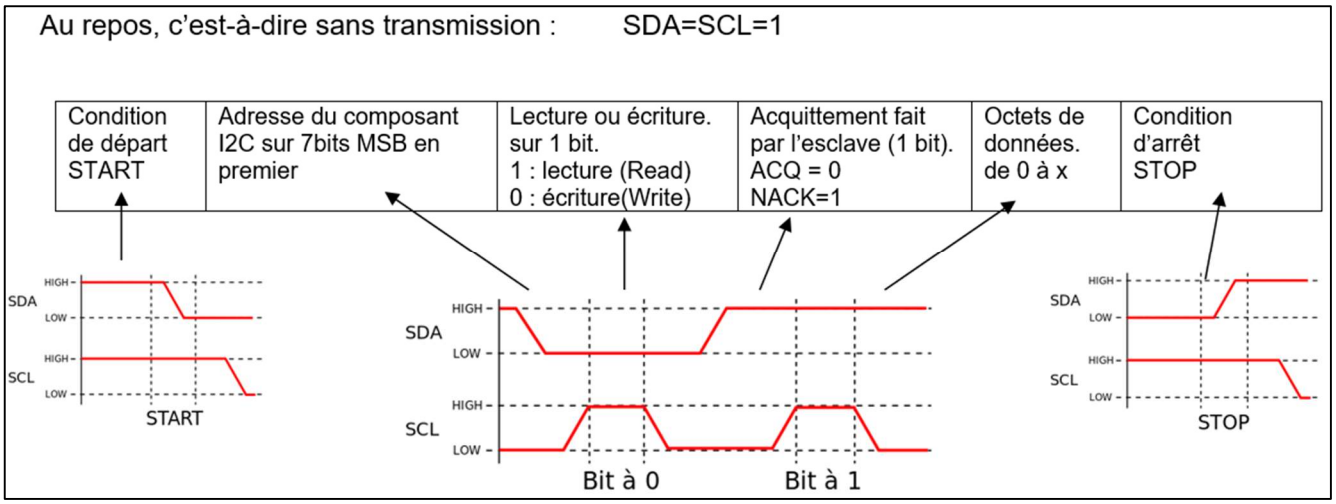
On a vu que le bus I2C utilise deux lignes très importantes : SDA et SCL.

Il est maintenant nécessaire de connaître comment utiliser SDA et SCL pour :

Commencer une transmission
Écrire des données (c'est le maître qui écrit)
Lire des données (c'est le maître qui demande des données à un esclave)
Finir une transmission

Forme des signaux pour SDA et SCL :

Start : Condition de départ
Écriture ou lecture d'un 0 ou 1 logique
Stop : Condition d'arrêt



Q4/ - A partir des signaux SDA et SCL, vus ci-dessus, déterminer :

-l'état repos (pas de transmission de données)

-une condition de start

-l'émission d'un 1 logique

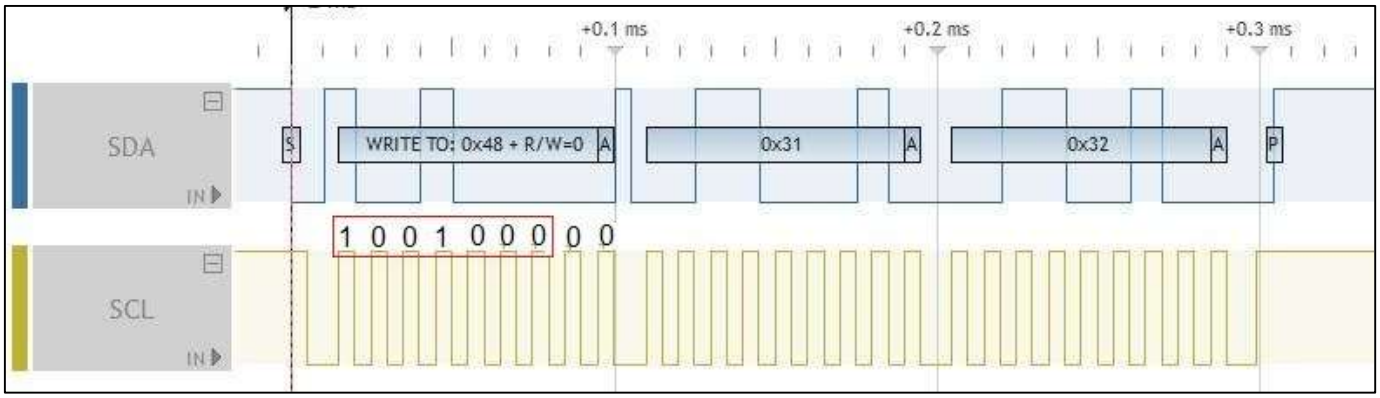
-l'émission d'un 0

-une condition de stop

Il existe un autre principe très important dans la transmission I2C :

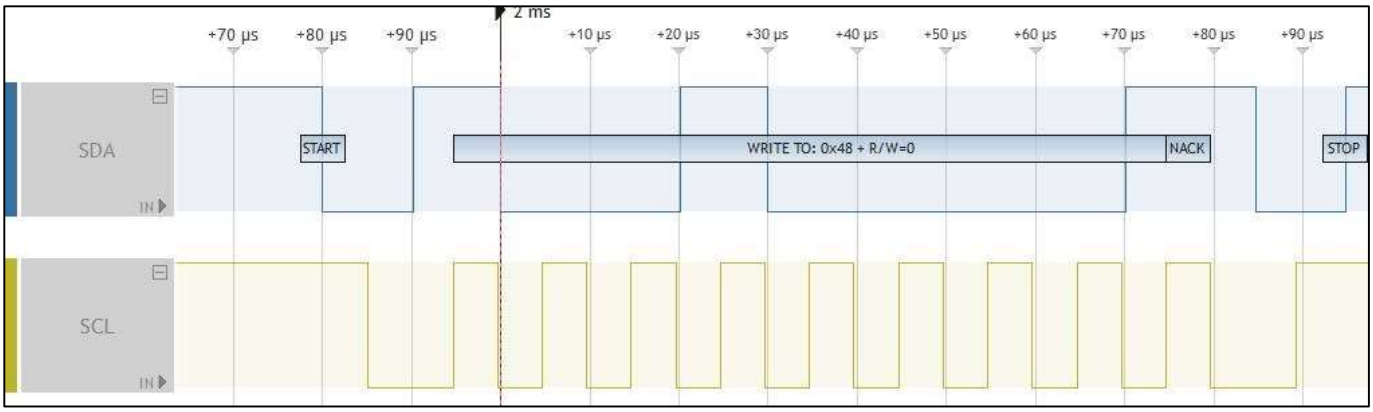
L'esclave doit toujours mettre à 0 logique le bit d'acquittement soit **ACK**.
Si l'esclave ne peut pas acquitter la réception d'un octet alors le bit d'acquittement restera à 1 soit **NACK**

Q5/ -Étudier la trame ci-dessus et donner le nombre de bits nécessaires pour écrire un octet sur le bus I2C.



Soit le chronogramme ci-dessus.

Q6/ -Que représente l'octet 0x48 ?
Donner le nombre d'octets de données, envoyé par le maître.



Soit le chronogramme ci-dessus.

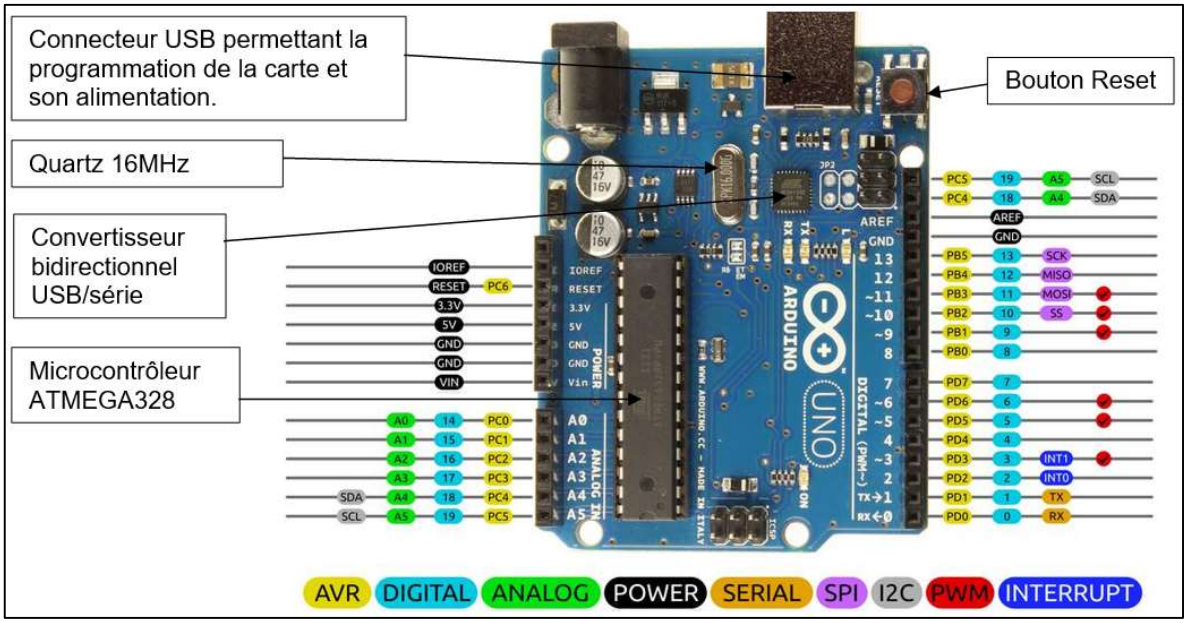
Q7/ -Que représente l'octet 0x48 ?

Quel est le niveau logique du bit d'acquittement ?

Quelle peut être la raison d'un **NACK** ? Justifier votre réponse.

Le bus I2C : Aspect logiciel

Présentation de la carte ARDUINO et des différentes fonctions pour programmer le bus I2C.



Aspect Logiciel :

1 Il faut faire un #include<Wire.h>
2 Les différentes fonctions possibles avec Arduino

Les plus utiles sont :

Wire.beginTransmission() ; // pour le maitre seulement
Wire.begin(0x48) ; // Mise en place d'un esclave (adresse : 0x48)
Wire.endtransmission() ; // Condition de stop

Pour écrire sur le bus I2C :

Wire.write(string data) ; // voir la doc pour les autres options possibles

Pour lire des données en provenance du bus I2C :

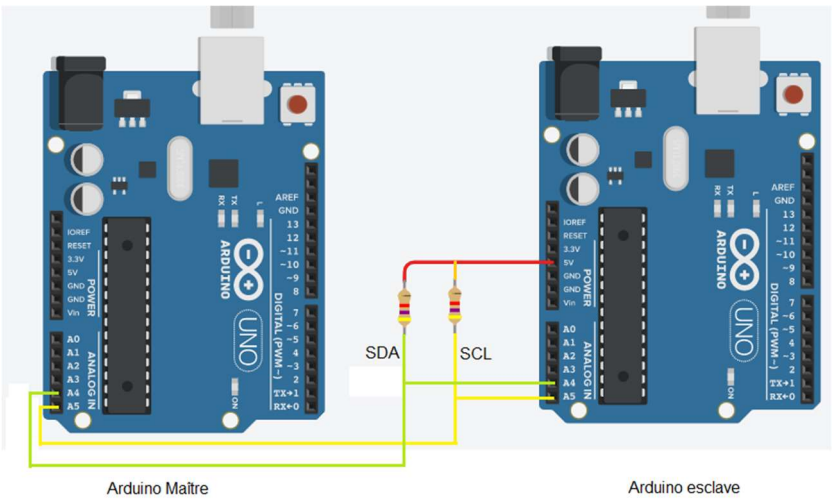
Byte data=Wire.read() ; associé avec Wire.available() ;

Functions

- begin()
- requestFrom()
- beginTransmission()
- endTransmission()
- write()
- available()
- read()
- onReceive()
- onRequest()

Q8/ - Donner le nom et le repère des broches associées au bus I2C.

Soit le schéma suivant pour le TP1 et TP2



TP1 : Le Maitre envoie des données à un esclave

Q9/ - Réaliser le montage ci-dessus avec une platine de tests. Faire vérifier.

Q10/ -Télécharger les deux fichiers ci-dessous et programmer les cartes Arduino.

Q11/ -Ouvrir le terminal de la partie esclave.

Quelles sont les données reçues ?

Donner l'adresse I2C du maître.

Donner l'adresse I2C de l'esclave. Justifier le résultat.

Q12/ -Modifier le ou les programmes pour afficher : TERM_SIN

TP2 : Le maître demande des données à un esclave

Q13/ -Télécharger les deux fichiers ci-dessous et programmer les cartes ARDUINO.

Q14/ -Ouvrir le terminal de la carte Arduino Maître. Justifier le résultat obtenu.

Q15/ - Donner l'adresse I2C de l'esclave.
Modifier les programmes pour afficher Hello dans le terminal du maître.

Mise en œuvre d'un capteur de température avec un protocole de type I2C.

Analyse du circuit DS1621

L'étude du cahier des charges a permis de choisir le DS1621 pour les raisons suivantes :

-le protocole de sortie est de type I2C comme l'afficheur LCD
-la gamme de mesures est très supérieure aux exigences : -55°C à +125°C
-précision de 0,5°C

Étude de la documentation du DS1621 : fichier PDF

Q16/ -Quelle est la plage de température mesurable par le DS1621 ?

Q17/ -Sur combien d'octets est codée l'information température ?

Si la température vaut 30°C, donner l'information numérique à obtenir ?

Si la température vaut 30.5°C, donner l'information numérique à obtenir ?

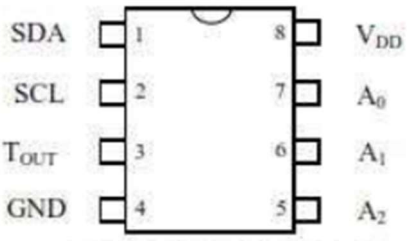
Étude de l'adresse du composant DS1621 :

On rappelle que le protocole I2C, utilise des adresses pour les circuits esclaves pour que le maître puisse les identifier et communiquer.

Les broches externes : A2, A1 et A0 permettent de choisir une adresse I2C pour le circuit DS1621.

A2, A1 et A0 sont numériques avec soit le niveau logique 0 ou 1.

L'adresse du circuit DS1621 est codée sur 7 bits.
Quand on code l'adresse I2C sur 8 bits, le bit de poids fort sera toujours à 0.



DS1621 8-PIN DIP (300mil)
See Mech Drawings Section

PIN DESCRIPTION

SDA	- 2-Wire Serial Data Input/Output
SCL	- 2-Wire Serial Clock
GND	- Ground
T_{OUT}	- Thermostat Output Signal
A0	- Chip Address Input
A1	- Chip Address Input
A2	- Chip Address Input
V_{DD}	- Power Supply Voltage

Codage de l'adresse I2C du circuit DS1621 sur 7 bits						
1	0	0	1	A2	A1	A0

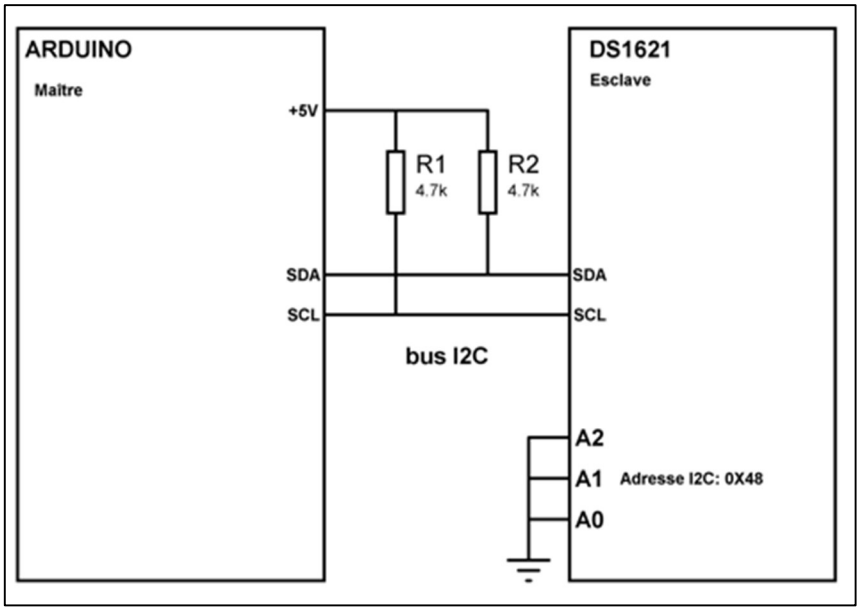
Q18/ -Donner le nombre d'adresses possibles (A2, A1 et A0) pour le DS1621.

Donner l'adresse sur 8 bits quand (A2 A1 A0 = 000) en base 10.

Donner l'adresse sur 8 bits quand (A2 A1 A0 = 111) en base 10

Quel est l'intérêt de pouvoir choisir l'adresse I2C de ce type de circuit ?

Schéma de câblage et mise en œuvre du DS1621



Le circuit DS1621 sera programmé via le bus I2C (carte ARDUINO maître) :

dans la fonction setup() :
le maître doit initialiser le capteur de température (DS1621) : choix du mode de fonctionnement.

dans la fonction loop() :
le maître va demander au capteur de température deux octets images de la température.

Programmation du DS1621 : Analyse des différents codes commande

Table 3. DS1621 COMMAND SET

INSTRUCTION	DESCRIPTION	PROTOCOL	2-WIRE BUS DATA AFTER ISSUING PROTOCOL	NOTES
TEMPERATURE CONVERSION COMMANDS				
Read Temperature	Read last converted temperature value from temperature register.	AAh	<read 2 bytes data>	
Read Counter	Reads value of count remaining from counter.	A8h	<read data>	
Read Slope	Reads value of the slope accumulator.	A9h	<read data>	
Start Convert T	Initiates temperature conversion.	EEh	idle	1
Stop Convert T	Halts temperature conversion.	22h	idle	1
THERMOSTAT COMMANDS				
Access TH	Reads or writes high temperature limit value into TH register.	A1h	<write data>	2
Access TL	Reads or writes low temperature limit value into TL register.	A2h	<write data>	2
Access Config	Reads or writes configuration data to configuration register.	ACh	<write data>	2

Le tableau ci-dessus montre que le DS1621 possède deux fonctions : thermomètre et thermostat.

Q19/ - Donner la valeur en hexadécimal correspondante au protocole (code fonction) suivant :

Access Config

Start Convert T

Read Temperature

On va maintenant mettre en œuvre la mesure de la température (DS1621) et réaliser un affichage dans le terminal.

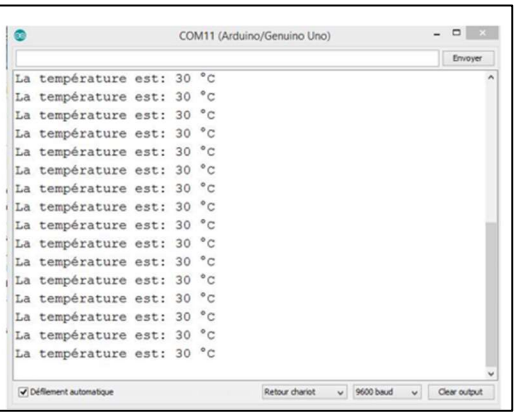
Dans le programme ci-dessus, il faut coder :

-la fonction INIT_DS1621()
-la fonction lect_DS1621()

Résultat à obtenir :

```
#include<Wire.h>
void setup() {
  Serial.begin(9600) ;
  init_DS1621() ;
}

void loop() {
  lect_DS1621() ;
  delay(500);
}
```



Q20/ -Télécharger le fichier et l'ouvrir avec le logiciel ARDUINO.

coder la fonction init_DS1621 en utilisant les informations ci-dessous.
finir la fonction lect_DS1621 en utilisant les commentaires proposés.

Fonction init_DS1621 : algo

1 Valider la liaison I2C.
2 Faire une condition de start et envoyer l'adresse I2C du DS1621.
3 Écrire la donnée I2C: 0xAC : on écrit dans le registre de configuration
4 Écrire la donnée I2C: 0x00 : conversion de température en continu
5 Condition d'arrêt (stop)
6 Attendre 20ms
7 Faire une condition de start et envoyer l'adresse I2C du DS1621
8 Écrire la donnée I2C 0xEE : lancement de la conversion en continu de la température
9 Condition d'arrêt (stop)

+

Fonction lect_DS1621 : A compléter

```
/*---------fonction lect_DS1621-----------------*/
void lect_DS1621()
{
 byte i=0;
 float tab[2];
 byte temperature;
 //Wire.beginTransmission(XX); // adresse DS1621
 //Wire.write(XX);             // AA (lancement conversion en continu
 Wire.endTransmission();     // fin transmission

// Wire.requestFrom(XX,XX);    // on attend 2 caracteres
 while(Wire.available()){    // caractère disponible sur I2C??
    tab[i]=Wire.read();      // lecture caractere et stockage dans tab
    i++;
  }
// if(tab[1]== XX)tab[0]=tab[0]+XX;
// temperature= ;
 Serial.println("la temperature est: ");
 Serial.println(temperature);
}
/*---------fin de la fonction lect_DS1621----------------*/
```