

Système embarqué : étude et réalisation d'un thermostat d'ambiance

Dispositif : durée 2h / salle informatique / travail par groupe

Prérequis

- Commande d'un actionneur, acquisition des données d'un capteur (SNT)
- IHM (SNT)
- modèle d'architecture VON NEUMANN : constituants d'une architecture à microprocesseur, micro-contrôleur (NSI)

Objectifs principaux

- Identifier le rôle des capteurs et actionneurs
- Réaliser par programmation une IHM répondant à un cahier des charges donné

Objectifs secondaires

- Repérer, dans un nouveau langage de programmation, les traits communs et les traits particuliers à ce langage
- Utiliser la documentation d'une bibliothèque

Cahier des charges

Le thermostat d'ambiance confère un confort incomparable grâce à sa précision et sa réactivité tout en allégeant la facture d'énergie. Le thermostat que vous allez réaliser devra respecter les spécifications suivantes :

- être capable de mesurer la température d'ambiance avec une précision de 0.1°C,
- maintenir la consigne de température avec une précision de +/- 1°C,
- la consigne de température sera réglable à partir d'une télécommande infrarouge,
- l'utilisateur visualise les valeurs de la température actuelle, la température de consigne, et l'état du système de chauffage (activé/ désactivé) sur un afficheur LCD.

Le matériel fourni

- une carte Arduino et son câble USB,
- une carte de connexion type breadboard et des fils de connexions,
- un capteur d'humidité et de température de type DHT11,
- un afficheur LCD 2 x 16 caractères rétro-éclairé sur bus I2C,
- un récepteur infrarouge vs1838b et sa télécommande,
- un module relais.

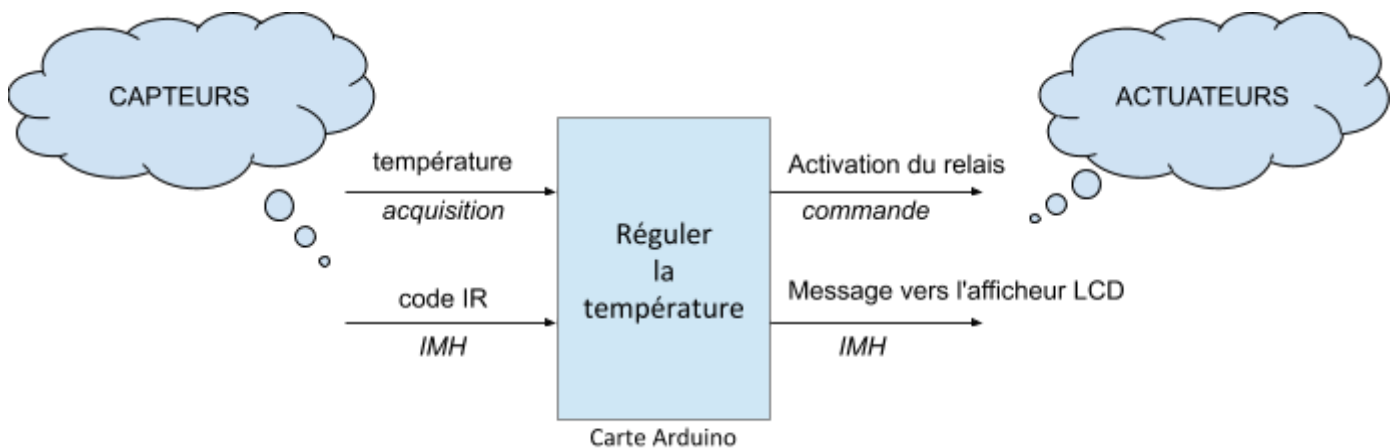
0 - Travail préliminaire : Caractérisation des signaux à traiter par la carte Arduino

a) Faire des recherches sur internet concernant le rôle de chaque composant du point de vue du micro-contrôleur, et cocher les cases correspondantes pour chaque composant relié à la carte Arduino :

Composant	Type de connexion		Type d'information		
	Entrée	Sortie	Analogique	Logique (TOR)	Numérique
Capteur DHT11	X (pin D2)				X (pseudo OneWire)
Récepteur IR VS1838B	X (pin D10)				X (serie 1 fil)
Relais		X (pin D4)		X	
Écran LCD		X (pins A4, A5)			X (I2C)

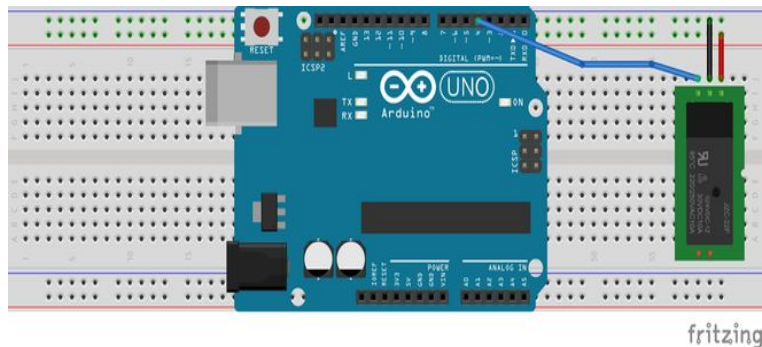
b) Quels composants contribuent à réaliser l'interface homme-machine (IHM) ?

L'écran LCD permet de visualiser l'état du système. La télécommande IR et son récepteur permettent d'agir sur la valeur de consigne.

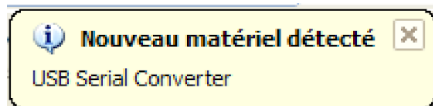


1 - Intégration du relais


Réaliser le câblage du relais à la carte Arduino à l'aide des câbles femelle/mâle conformément à la figure suivante (l'entrée de commande **IN** du relais est reliée sur l'entrée **D4** de l'Arduino) :

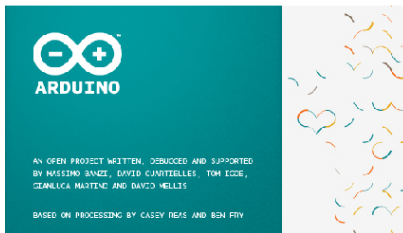


Relier la platine Arduino à l'ordinateur à l'aide du cordon USB fourni.



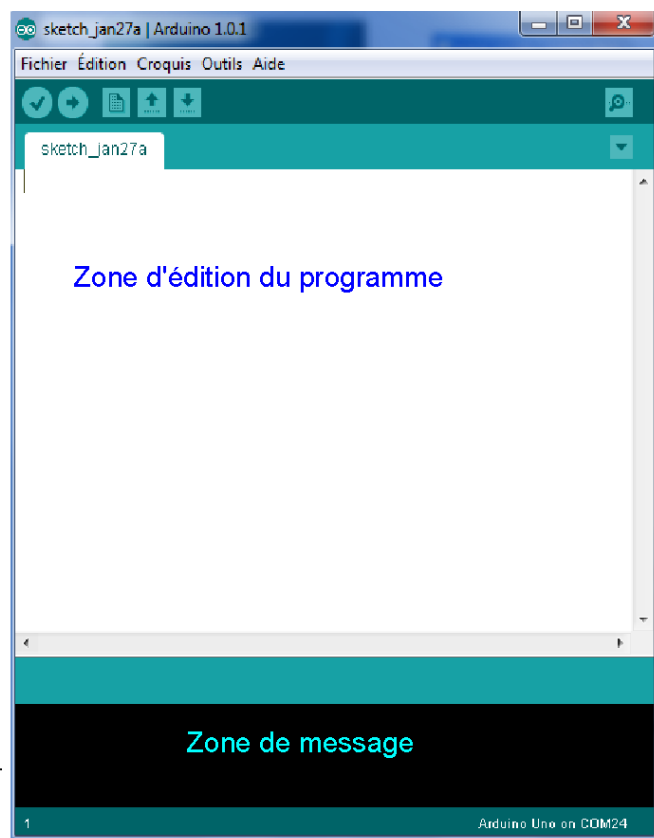
Attendre après la détection par le système, que le nouveau périphérique soit prêt à fonctionner.

Lancer l'application sur le PC, soit en cliquant sur l'icône  présente sur le bureau, soit en lançant le programme depuis le menu démarrer



La zone d'édition permet de saisir
Le programme en langage C,

La zone de message permet
de lire les messages envoyés par
le logiciel suite à une opération.



Dans la zone d'édition, tapez le code suivant :

```
#define relay_pin 4 // set D4 as pin Relay

void setup() {
    pinMode(relay_pin, OUTPUT); // set pin relay as output
}

void loop() {
    digitalWrite(relay_pin, HIGH);
    delay(5000);
    digitalWrite(relay_pin, LOW);
    delay(5000);
}
```

- Comment déclare t-on une constante nommée en C ?
- Que signifie le type *void* ? Quels semblent être les rôles des procédures *setup()* et *loop()* ?
- Compiler et téléverser le programme sur la carte Arduino. Que fait t-il ?

2 - Intégration du capteur DHT11

INSÉRER FIGURE DE CABLAGE AFFICHEUR ICI

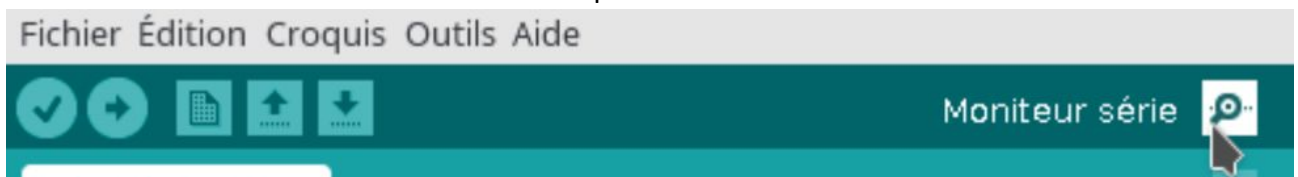
- Relier le capteur DHT11 comme sur la figure ci-dessus.
- Dans le menu Fichier\Exemples\DHT11, ouvrir le sketch **DHTTester**
modifier le code pour l'adapter à votre configuration

```
#define DHTPIN 4 // Digital pin connected to the DHT sensor
// Feather HUZZAH ESP8266 note: use pins 3, 4, 5, 12, 13 or 14 --
// Pin 15 can work but DHT must be disconnected during program upload.

// Uncomment whatever type you're using!
#define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
// #define DHTTYPE DHT21 // DHT 21 (AM2301)
```

Comment écrit-on des commentaires ?

- Téléverser le code et ouvrir le moniteur série pour observer le résultat.



- Identifier et expliquer la ligne de code permettant de récupérer l'information température

```
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
:
```

En quoi la déclaration de variable est-elle différente de celle en Python ?
Le typage est statique et non plus dynamique.

e) Quelles lignes de code sont indispensables pour utiliser le capteur de température ?

En début de code :

```
#include <DHT.h>

#define dht_pin 4
#define dht_type DHT11

DHT dht(dht_pin, dht_type);
```

Dans la fonction void setup():

```
dht.begin();
```

f) **(en collectif)** Proposer un algorithme “naïf” qui compare la température ambiante avec celle de consigne et déclenche le relais de chauffage en conséquence.

```
#include <DHT.h>

#define relay_pin 4 // set D4 as output Relay pin
#define dht_pin 2 // set D2 as input Temp pin
#define dht_type DHT11

DHT dht(dht_pin,dht_type);

float t_cons = 21;
float t_amb;

void setup() {
  pinMode(relay_pin, OUTPUT);
  digitalWrite(relay_pin, LOW);
}

void loop() {
  t_amb = dht.readTemperature();
  if(t_amb<t_cons) {
    digitalWrite(relay_pin, HIGH);
  }
  else {
    digitalWrite(relay_pin, LOW);
  }
}
```

g) Lancer ce code en fixant un valeur de température proche de la température ambiante. En

manipulant le capteur de température, que constate-t-on sur le fonctionnement du relais ?
Le relais change d'état à une fréquence rapide.

- h) Modifier le code pour qu'il tienne compte de la différence de température entre celle de consigne et celle ambiante.

Le vérifier et le déboguer en insérant des messages vers le moniteur série si nécessaire :

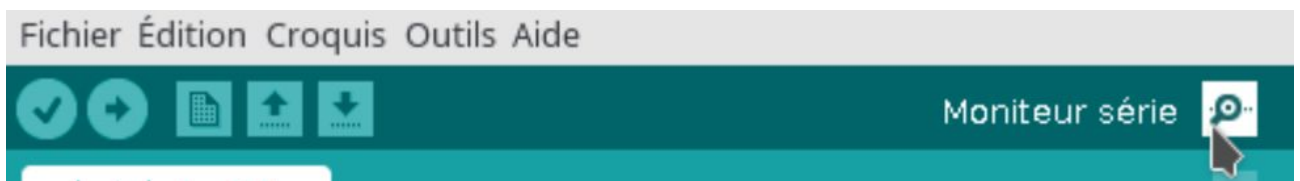
Dans la procédure void setup():

`Serial.begin(9600);`

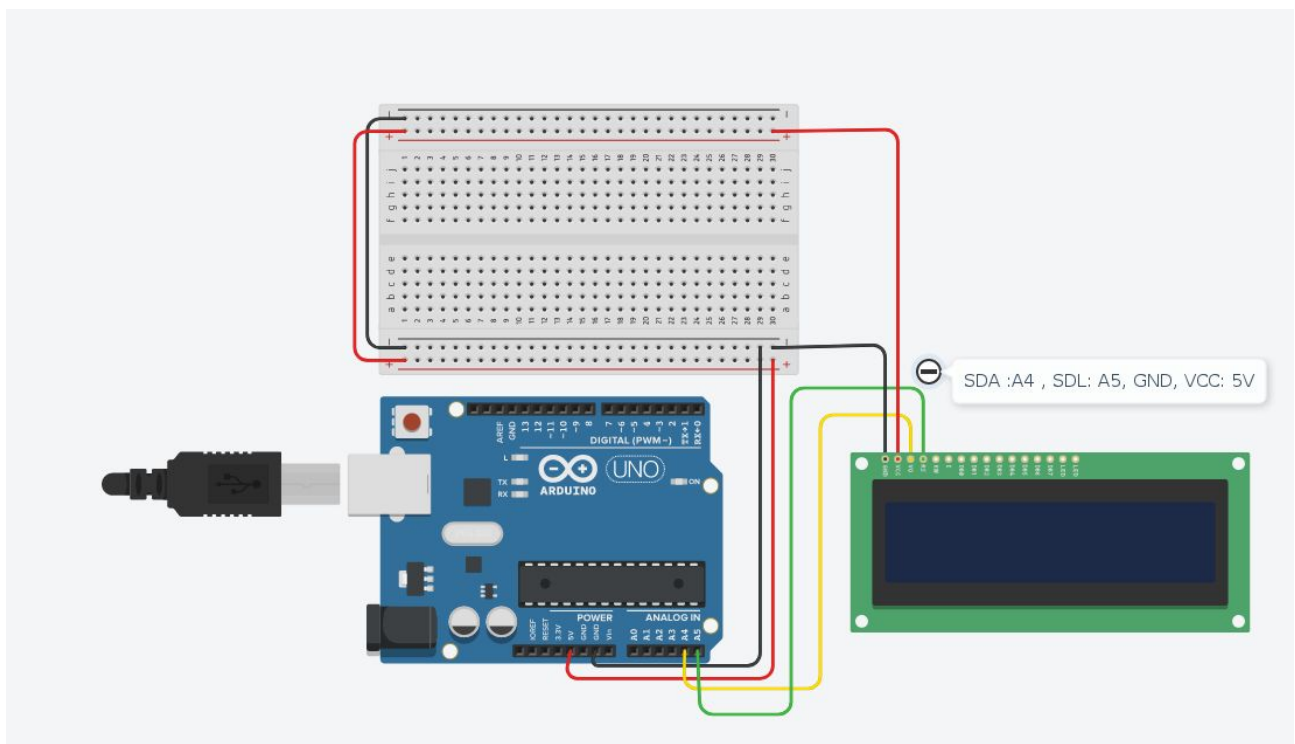
Dans la procédure loop setup():

`Serial.println("message à afficher")` ou `Serial.print("message sans retour à la ligne")`

Puis utiliser le "Moniteur série" avec vitesse fixée à 9600 baud.



3 - Intégration de l'afficheur



- a) Câbler l'afficheur LCD conformément à la figure
- b) Dans le menu Fichier\Exemples\LiquidCrystal_I2C, ouvrir le sketch **HelloWorld**

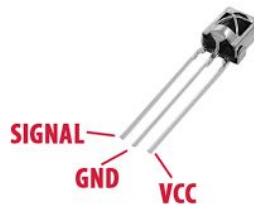
- c) L'écran LCD utilisé est un **2 lignes de 16 caractères** : modifier la ligne de code suivante pour intégrer la particularité de votre écran

```
LiquidCrystal_I2C lcd(0x27,16,2);
```

La fonction `lcd.setCursor(1,0)` : permet de positionner le curseur sur la 1ere colonne de la première ligne indexée à 0.

- d) Utiliser cette instruction dans le code d'exemple de manière à écrire :
"Thermostat V1" centré sur la première ligne, et "Bienvenue" centré sur la deuxième ligne.
- e) Intégrer les lignes nécessaires dans le code précédemment produit afin d'avoir le message de bienvenue à l'allumage du système.
- f) Intégrer le code nécessaire afin d'afficher la température ambiante "Ambiance : XX.XX" sur la ligne 1 alignée à gauche, la température de consigne et l'état du relais "C: YY.YY E:ZZZ" alignés à gauche sur la ligne 2 où ZZZ est l'état "ON " ou "OFF".

4 - Intégration du récepteur IR



INSÉRER FIGURE DE CABLAGE AFFICHEUR ICI

- a) Relier le récepteur IR comme sur la figure ci-dessus
- b) Ouvrir le sketch fourni **ircode** .

Téléverser le code dans la carte, et ouvrir la console.

Appuyer sur les touches de la télécommande fournie, et repérer et noter à l'aide de la console, les codes hexadécimaux correspondant.

Choisir les touches et les codes associés qui vous permettront d'augmenter et diminuer la valeur de la température de consigne.

- c) Ouvrir le sketch **thermostat** et insérer les codes hexadécimaux correspondant à vos choix, téléverser le code, et vérifier le bon fonctionnement.