Méthodes numériques

Microcontroleur Arduino

Pour davantage de détails, d'explications et d'exemples, on pourra aller sur la page suivante :
https://phychim.ac-versailles.fr/spip.php?article1076

Pour accéder aux pages suivantes de ce poly, et pour cliquer sur les liens, ouvrir la version pdf (site de la classe, rubrique TP-DS-maths).

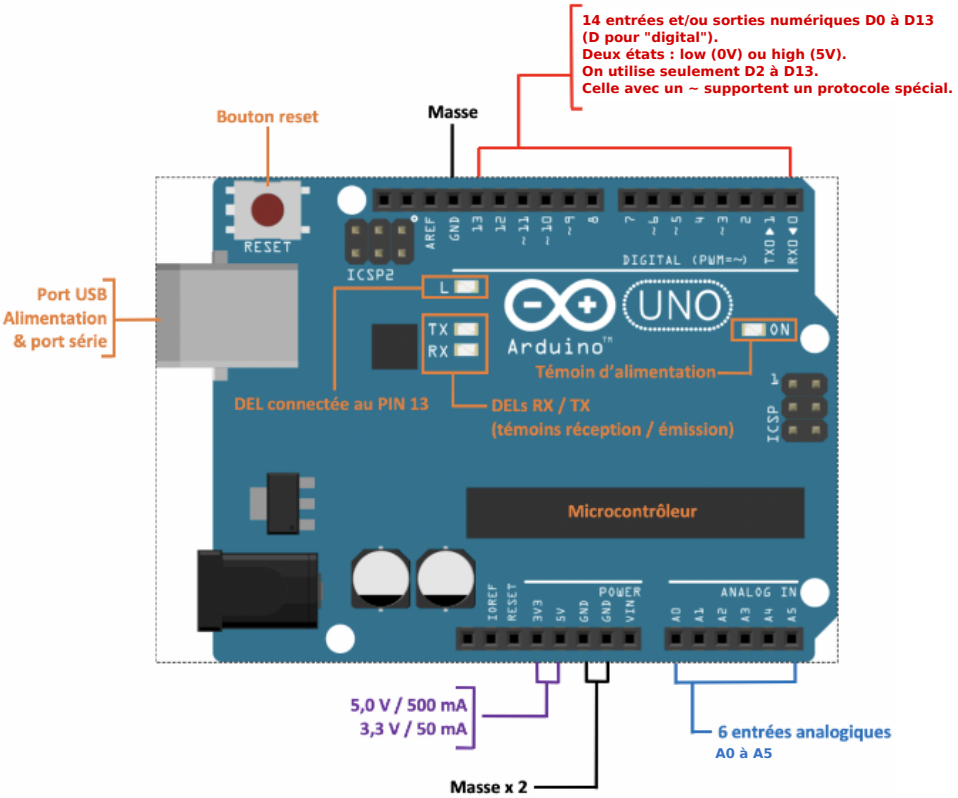
I Généralités

I.1 Structure de la carte

Un microcontroleur permet de prendre en entrée des signaux numériques ou analogiques (en provenance de capteurs, de boutons de commande, d'un autre microcontroleur...), de les traiter à l'aide d'un programme défini par l'utilisateur, puis de produire des sorties (tensions ou courants qui permettent de commander des actionneurs, des LED..., ou encore messages à destination de l'utilisateur).

Une carte Arduino (ici modèle Arduino UNO) est un microcontroleur. Les éléments principaux sont :

- Le port USB : permet à la fois l'alimentation et la communication avec l'ordinateur.
- Une borne d'alimentation : en cas de fonctionnement autonome sans le cable USB.
- Bouton reset : permet de refaire démarrer le programme présent sur la carte.
- Des LED témoins.
- Des broches GND (ground) : il s'agit de la masse (0 V).
- Une broche 5 V et une 3,3 V : permet l'alimentation des capteurs, des LED, etc.



On parle de broche, de "pin" ou de "port" pour désigner les entrées ou sorties.

- **Les entrées analogiques A0 à A5 :** elles lisent une tension comprise entre 0 et 5 V, et la convertissent en un entier compris entre 0 et 1023.

  Exemple pour lire l'entrée A0 : `float tension = 5./1023.*analogRead(pinCapteur);`

- **Les entrées et/ou sorties numériques D0 à D13.** Elles prennent uniquement les valeurs 0 et 5 V.
  - Mettre le port D4 en mode sortie : `pinMode(4, OUTPUT);`
    On peut alors le mettre à 5 V avec `digitalWrite(4, HIGH);` et à 0 V avec `digitalWrite(4, LOW);`
  - Le mettre en mode entrée : `pinMode(4, INPUT);`
    On peut alors lire son état avec `int valeur = digitalRead(4);` Ainsi `valeur` contient soit `HIGH` soit `LOW`.
- Il y a d'autres ports. Voir documentation en ligne.

Remarque : le port série ou moniteur série sert à afficher, sur l'ordinateur, des chaines de caractères en provenance de la carte. Cf plus loin.

I.2 Écriture et téléversement du programme depuis l'ordinateur

Le programme qui dit quoi faire au microcontroleur doit être écrit sur l'ordinateur à l'aide d'une interface dédiée (soit un logiciel Arduino à installer à partir du site https://create.arduino.cc, soit une interface web https://create.arduino.cc/editor).

Les symboles ci-dessous sont importants :

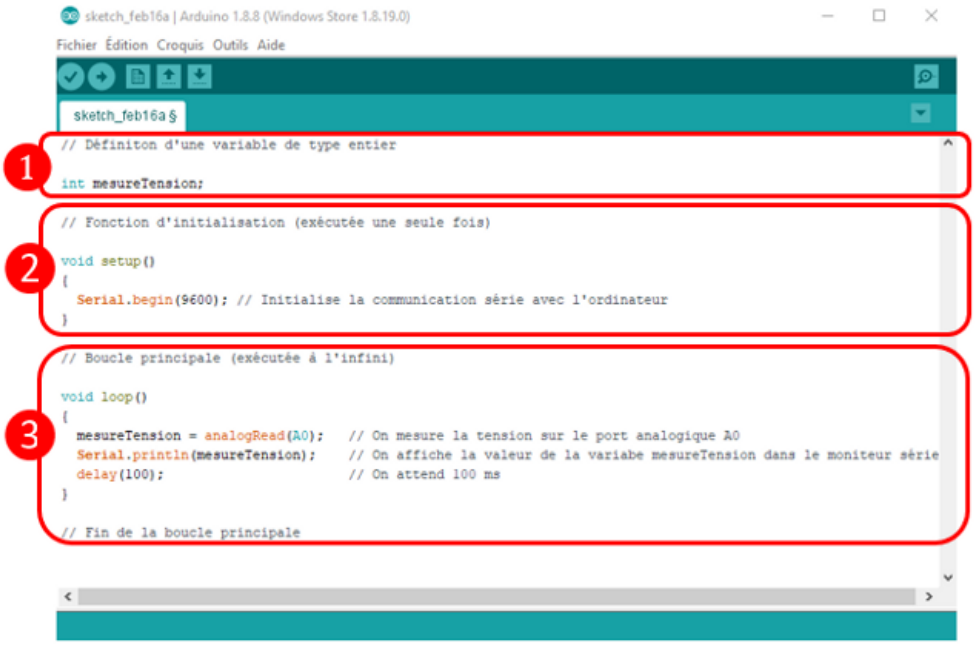
- Vérifier : compile le programme et en vérifie la syntaxe.
- Téléverser : transfert le code compilé sur la carte.
- Moniteur série : ouvre la fenêtre où s'affichent les résultats des commandes de type `print`.

Attention : au préalable, il faut avoir choisi le bon type de carte dans `outils -> type de carte` (ici c'est une Arduino UNO), ET dans `outils -> port` il faut sélectionner le port COM sur lequel la carte apparaît.

Une fois que le programme est téléversé sur la carte, celle-ci l'exécute en permanence. La structure d'un programme est toujours la même :

Ci-contre un exemple qui mesure la tension sur le port A0 puis qui l'affiche dans le moniteur série (sur l'ordinateur).

- 1 : les variables doivent être déclarées avant utilisation. Dans ce bloc on déclare également les bibliothèques, etc.
- La fonction `setup()` est exécutée une fois au lancement du programme.
- La fonction `loop()` est exécutée en boucle à l'infini.



I.3 Le langage de programmation

Le langage de programmation utilisé est spécifique. On retrouve les instructions classiques (if, then, for, while, ...).

Attention par rapport à Python : chaque instruction doit finir par un point virgule.

Quelques liens pour approfondir :

- Référence du langage : https://www.arduino.cc/reference/fr/
- Exemples simples : https://phychim.ac-versailles.fr/spip.php?article1076

I.4 Simuler une carte Arduino

Si on ne dispose pas d'une carte Arduino "en vrai", on peut en simuler une en ligne. Par exemple sur le site https://www.tinkercad.com/ (tutoriel d'explication en vidéo à la fin du lien donné au tout début du document).

Les trois "exemples de prise en main" qui sont dans la suite de ce poly seront aussi disponibles en version simulée ici.

Remarque : dans la liste des composants, "kit de démarrage" puis "Arduino", permet de charger plusieurs exemples de base. Pour accéder au code cliquer sur `code`. Prendre alors "mode d'édition : texte".

II Exemples de prise en main

Matériel par groupe pour les trois exemples (cf photos) :

- Carte Arduino,
- thermistance CTN et ensemble "pont diviseur de tension",
- émetteur/récepteur ultrason.

Pour la classe : un thermomètre, un multimètre et deux fils, si possible quelques diodes et résistances 220 Ω (pour mettre en série avec les diodes), et quelques platines d'essai + fils.

Remarque : les cartes du lycée sont montées avec une "surcouche" qui vient par dessus la carte d'origine et qui ajoute des connexions assez pratiques vers les différents ports (ces connexions sont des raccourcis, elles ne sont pas nouvelles). Ce qui est sur la première page du poly est donc la carte d'origine, sans cette surcouche.

II.1 Faire clignoter la LED du port D13

Sur les cartes Arduino UNO, il y a une LED qui est liée au port digital D13. Le programme suivant la fait clignoter. Il servira surtout à comprendre la structure de base d'un programme, et à voir si la carte est bien reliée et reconnue par l'ordinateur.

Il n'y a aucun cablage à effectuer, à part brancher la carte au port USB.

```
const int LED_port = 13;  // sur Arduino UNO, la LED est reliée au port digital (ou numérique) 13

// Boucle d'initialisation (exécutée une seule fois)
void setup() {
  pinMode(LED_port, OUTPUT);      // déclare le port numérique D13 comme une sortie
}

// Boucle principale (exécutée à l'infini)
void loop() {
  digitalWrite(LED_port, HIGH);   // passe le port D13 à la valeur haute (5V)
  delay(1000);                    // attends 1000ms
  digitalWrite(LED_port, LOW);    //  passe le port D13 à la valeur basse (0V)
  delay(200);                     // attends 200ms
}
```

1. Ouvrir le logiciel Arduino sur l'ordinateur.
2. Aller sur https://www.tinkercad.com/things/iaKzDjPBBdE pour récupérer le code. Pour le voir, deux options :
   - Cliquer sur `simuler`, puis sur `code`.
   - Ou bien cliquer sur `S'inscrire pour copier` (nécessite de s'inscrire, pourquoi pas car vous pourrez alors tester et éditer en ligne les exemples). Cliquer alors sur `code`.

   Copier alors l'ensemble du code dans le logiciel Arduino sur l'ordinateur.
3. Brancher la carte Arduino sur le port USB de l'ordinateur. Dans `outils -> port` du logiciel, sélectionner le port COM sur lequel la carte apparaît.
4. Cliquer sur `valider`, puis sur `téléverser`. Si tout va bien, il n'y a pas d'erreurs et la LED de la carte clignote ! (pas la grosse LED verte, mais une petite LED à côté du port USB)
5. Essayer de modifier le code pour changer la fréquence de clignotage de la LED.

II.2 Capteur de température

Il existe de multiples capteurs de température. On utilise ici un capteur CTN : il s'agit d'une résistance dont la valeur varie en fonction de la température T. Ainsi mesurer R permet, à l'aide de la formule suivante, d'en déduire T :

$$T = \frac{1}{\frac{1}{T_0} + B\frac{1}{B}\ln\frac{R}{R_0}},$$

avec pour la CTN utilisée les valeurs $T_0 = 298{,}15\,\text{K}$, $R_0 = 100\,\text{k}\Omega$, $B = 4250\,\text{K}$.

Pour mesurer cette résistance R, on mesure la tension u au milieu d'un pont diviseur de tension avec une résistance R_1. On a alors

$$u = U_0\frac{R_1}{R_1 + R}, \quad \text{soit en isolant } R \text{ :} \quad R = R_0\left(\frac{U_0}{U} - 1\right).$$

Cette tension u est lue sur un port analogique : `tension = 5./1023.*analogRead(pinCapteur);`.

Ce code affiche la température sur le port série, c'est-à-dire sur l'ordinateur. Il faut donc l'ouvrir (`outils -> moniteur série`, ou bien icone en haut à droite).

1. Pour obtenir le code : https://www.tinkercad.com/things/hKw2FouiRqN et procéder comme dans l'exemple 1. Copier-coller ce code dans un nouveau fichier du logiciel Arduino, sur l'ordinateur.

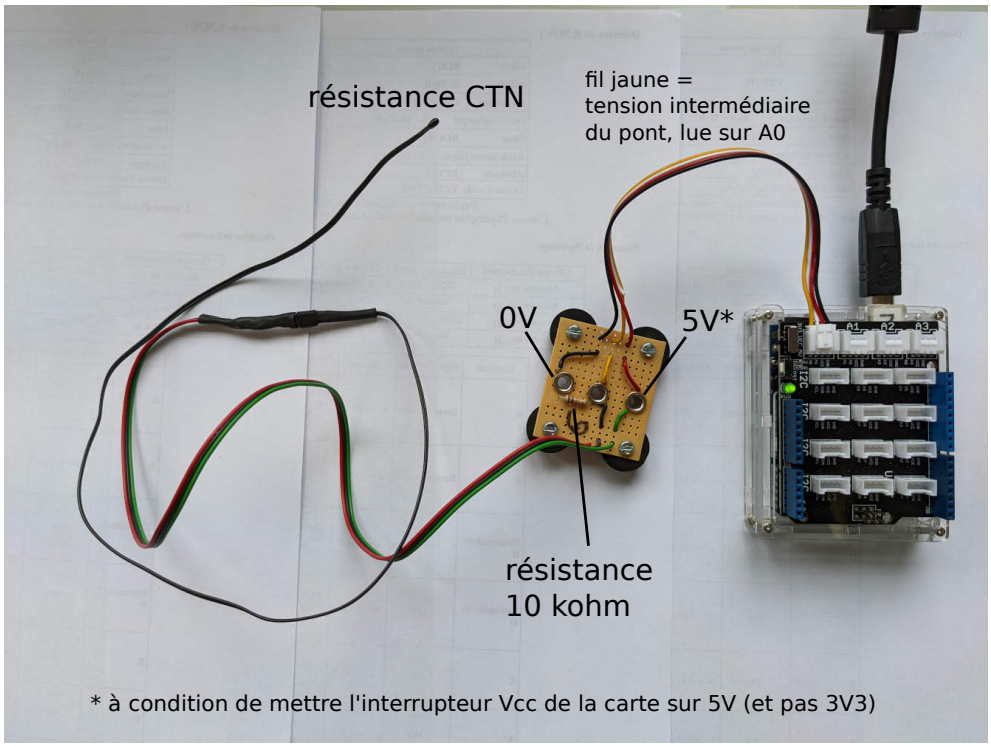
   Attention, le code sur tinkercad est écrit pour utiliser un photorésistor à la place de la thermistance CTN (car celle-ci n'existe pas dans ce logiciel en ligne). Il faut donc décommenter les 4 lignes qui concernent la température.

2. Réaliser les branchements sur la carte. Il y a juste à connecter la broche du capteur sur la broche où il y a A0.

3. Vérifier et téléverser le code. Ouvrir le moniteur série et voir si la température affichée semble correcte.

4. Modifier le code pour qu'il affiche, à coté de la température, les mots "il faut chaud" si $T \geq 25\,°\text{C}$.

   On peut également chercher à allumer une diode lorsque $T \geq 25\,°\text{C}$...




```arduino
#include <math.h>            // pour avoir le log

const int B = 4250;          // constante B de la CTN
const float R0 = 10000;      // valeur de la résistance CTN à 25 C
const float R1 = 10000;      // résistance du pont diviseur
const int pinCapteur = A0;   // broche de connexion du capteur

void setup()
{
    Serial.begin(9600);  // "Port série" : sert à écrire des résultats sur l'ordinateur avec Serial.print()
}
```

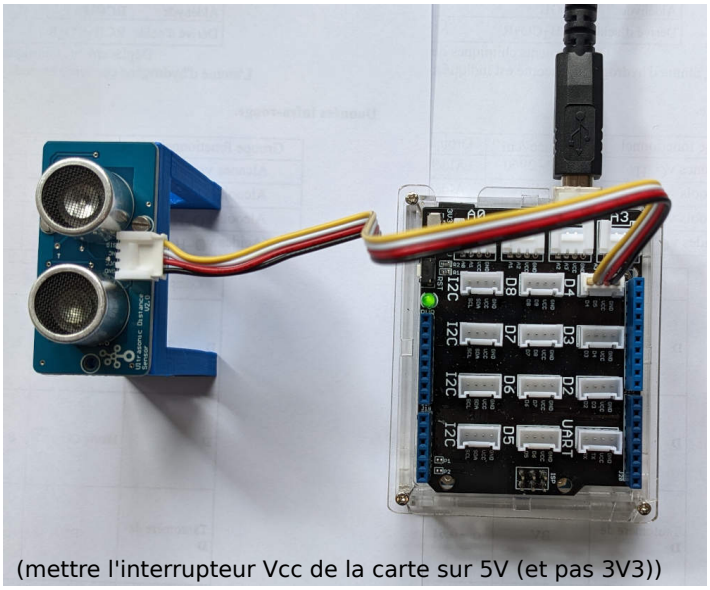
```
void loop()
{
    float tension = 5./1023.*analogRead(pinCapteur); // lecture de la tension au milieu du pont
    float R = R1*(5./tension - 1.0);    // conversion en valeur de la résistance du capteur (pont diviseur)
    float temperature = 1.0/(log(R/R0)/B+1/298.15)-273.15;  // passage de la valeur de R à celle de T (°C)

    // affichage dans le moniteur série
    Serial.print("tension pont = ");
    Serial.print(tension);
    Serial.print(" V");
    Serial.print("\t"); // tabulation
    Serial.print("resistance capteur = ");
    Serial.print(R);
    Serial.print(" ohm");
    Serial.print("\t"); // tabulation
    Serial.print("temperature = ");
    Serial.print(temperature);
    Serial.print(" C");
    Serial.print("\n"); // saut de ligne
    delay(1000);  // attente de 1000ms avant de recommencer
}
```

II.3 Mesure de distance avec un capteur ultrason

On utilise ici un émetteur/récepteur d'ultrasons. Le principe est le suivant :

- Envoi d'un pulse (signal à 40 kHz) d'une largeur de 10 µs.
- Mesure du temps d'aller-retour de ce pulse.

1. Pour obtenir le code : https://www.tinkercad.com/things/dZjSGsXaZ4q et procéder comme dans l'exemple 1. Copier-coller ce code dans un nouveau fichier du logiciel Arduino, sur l'ordinateur.
2. Réaliser les branchements nécessaires sur la carte. Normalement il y a juste à connecter le cable du capteur sur la carte, sur la broche où il y a le port D4.
3. Vérifier et téléverser le code. Ouvrir le moniteur série et voir si le capteur de distance fonctionne.
4. Modifier le code pour qu'il affiche, à coté de la distance, les mots "c'est loin !" si $d \geq 1\,\text{m}$.
   On peut également chercher à allumer une diode lorsque $d \geq 1\,\text{m}$...



(mettre l'interrupteur Vcc de la carte sur 5V (et pas 3V3))

```
// variables :
int pin_capteur = 4;     // broche où est l'ER ultrasons (ici D4)
float distance;
int chrono;
float celerite = 347;    // vitesse du son (m/s)

void setup()
{
Serial.begin(9600);      // initialise communication avec moniteur série
}

void loop()
{
// émission d'un pulse de 10 µs
pinMode(pin_capteur, OUTPUT);      // broche mise en mode sortie (émission)
digitalWrite(pin_capteur, HIGH);   // broche niveau haut : émission
delayMicroseconds(10);             // délai de 10 µs
digitalWrite(pin_capteur, LOW);    // broche niveau bas : fin d'émission

// le module ultrason va envoyer un signal HIGH dès qu'il reçoit un retour
pinMode(pin_capteur, INPUT);          // broche mise en mode entrée, en attente du module
chrono = pulseIn(pin_capteur, HIGH); // chronométrage du temps avant réception du signal (micro s)

// calculs
distance = celerite*chrono/1000000/2; // calcul de la distance (m)

// affichage
Serial.print("Duree a/r : ");
Serial.print(chrono);
Serial.print(" micro s \t");
Serial.print("Distance : ");
Serial.print(distance);
Serial.println(" m");
delay(1000);                          // délai de 1000 ms avant la mesure suivante
}
```