*Programmation* **objet**

TP4/5 : Arduino/ Processing

But : Création d’une interface graphique sous Processing contrôlée par le récepteur IR de l’Arduino via une télécommande

**PARTIE 1:**

0] Introduction

Il vous est demandé de créer un programme Arduino / Processing permettant de :

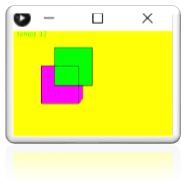
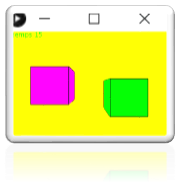
Pour la partie Arduino :

-Utiliser une télécommande de télévision comme une manette “haut bas gauche droite”.

Pour la partie Processing :

-Créer une interface graphique dont l’affichage est composé de 2 cubes de couleurs différentes. Le premier cube (cube 1) doit être contrôlé dans le plan par la télécommande. Le deuxième cube (cube 2) est qualifié de cible. Dès que le cube 1 est “superposé” sur le cube 2 par l’utilisateur, le cube 2 change de coordonnées et le nombre de points est incrémenté de 1.

Exemple:



I] Code Arduino

Le code Arduino doit permettre de :

* Créer une liaison série et activer le récepteur IR (setup),
* Réceptionner le résultat (adresse de la touche de la télécommande),
* Afficher le résultat sur le moniteur série.

1. *Utilisation de la bibliothèque IRremote.h*

Éléments utiles :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bibliothèque | Classe | Objet (constructeur) | Fonction | Variables |
| IRremote.h | **IRrecv** | *Mon\_objet\_IR*(num Pin) | *Mon\_objet\_IR*.**enableIRIn**() |  |
| *Mon\_objet\_IR*.**decode**(*Mon\_objet\_résultat*) |
| *Mon\_objet\_IR*.**resume**() |
| **decode\_results** | *Mon\_objet\_résultat* |  | Mon\_objet\_résultat.  **value** |

**enableIRIn**() : Active le récepteur infrarouge

**decode**(*Mon\_objet\_résultat*) : Réceptionne l’information émise

Mon\_objet\_résultat.**value :** Valeur de “Mon\_objet\_résultat”

*Mon\_objet\_IR*.**resume**() : “prépare” le prochain code

*Aide : Voir l’annexe 2 pour connaître la PIN du récepteur IR*

*B) Test du code avec une télécommande*

Vérifier que le code fonctionne avec la télécommande.

A l’aide du moniteur série, afficher et mémoriser les codes concernant les touches “haut, bas, gauche, droite “ de la télécommande.

II] Code Processing

Le code Processing doit permettre de :

* Créer une interface graphique 3D de taille 300, 200 composée d’un cube “contrôlable” (Cube CO) et d’un cube “cible” de position aléatoire (Cube CI),
* Communiquer en 9600 sur le port série de l’Arduino NANO,
* Arrêter le programme lorsque le temps est écoulé.

**Etape 1 : Création des classes et affichage des objets**

Deux classes sont à créer : la classe « Cible » et la classe « Cube ». Les deux classes sont identiques mise à part l’initialisation des couleurs (rose pour le cube, vert pour la cible) (voir cours). Elles auront comme membres des coordonnées dans l’espace (x,y,z) et une fonction membre « affichage ».

* Graphiquement, le « cube » et la cible auront comme valeurs constantes (longueur, largeur, épaisseur):

int L = 50;

int l = 50;

int e = 50;

* Pour le déplacement, le « cube » aura les valeurs initiales ci-dessous et se déplacera uniquement dans le plan (le code du déplacement se fera en Etape 2):

int x0 = 125;

int y0 = 75;

* Pour l’initialisation aléatoire, la cible aura comme valeurs :

Int xc = int(random(0,300-l));

Int yc = int(random(0,200-L));

Pour l’étape 1, faire seulement une seule initialisation des cibles, au moment du setup.

**Etape 2 : Déplacement du cube**

Afin de déplacer le cube, une fonction « reception » doit être créée.

Cette fonction permettra de :

* Vérifier la connexion du port. while (port.available() > 0)
* Tant que la connexion fonctionne, récupérer les données lues sur le port : String inByte = port.readString();
* Vérifier les commandes reçues : exemple String[] verificationD = match(inByte, "1074794287"); pour la commande droite.
* En fonction du résultat, modifier les valeurs de x0 et y0 d’un pas de 1 (décalage d’une unité à droite/gauche/haut/bas)

**Etape 3 : Vérification et fin**

Afin de vérifier que :

* Soit le cube superpose la cible
* Soit le temps est écoulé

Il faut :

Créer une fonction vérification qui :

* met fin au programme lorsque le temps est écoulé
  + s = millis()/1000
* rappelle la fonction « initialise cible » lorsque le cube superpose la cible

**PARTIE 2**

*But : Contrôler le CUBE CO via une télécommande afin de le superposer au CUBE CI ayant la même couleur que la LED allumée sur la carte ARDUINO*

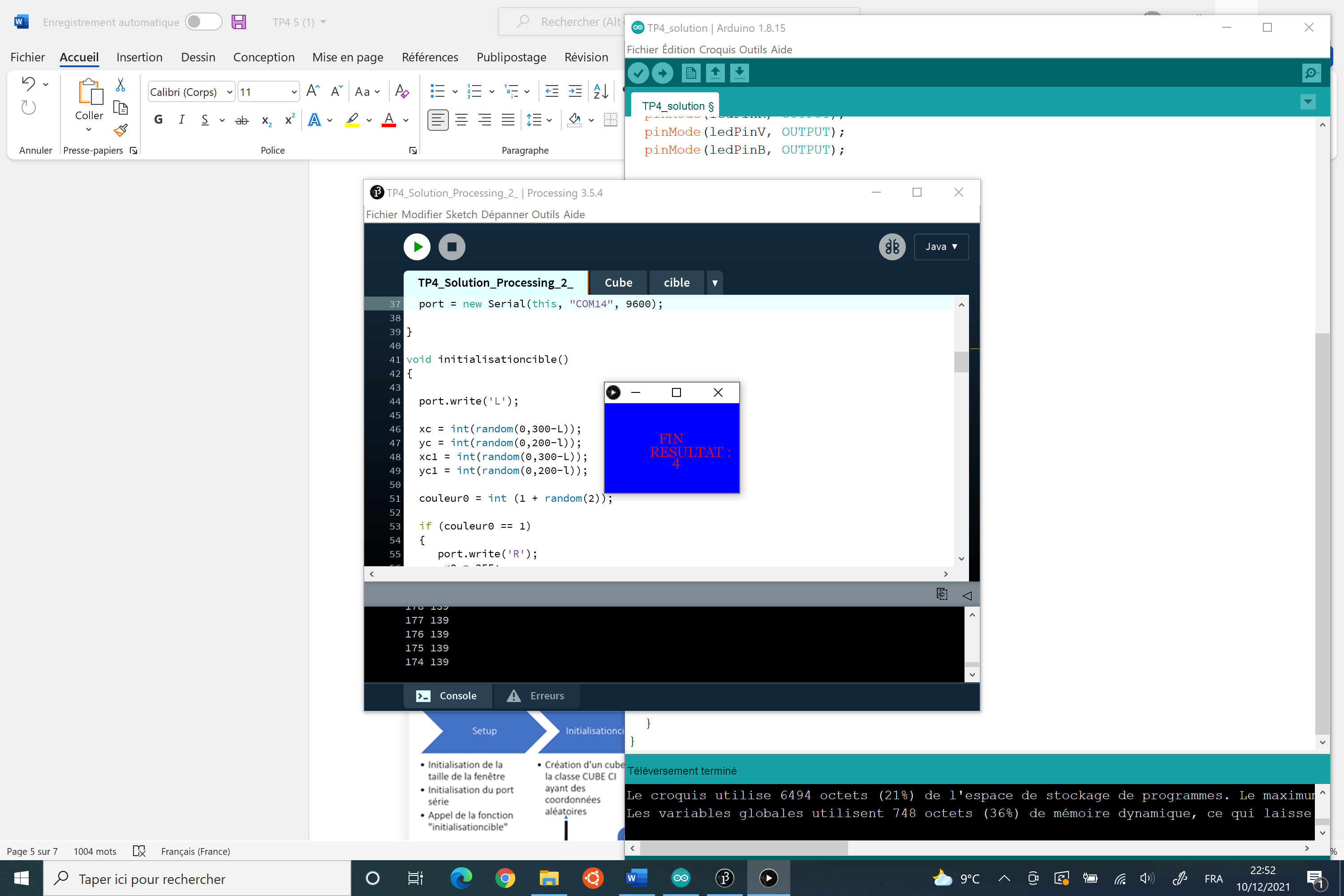
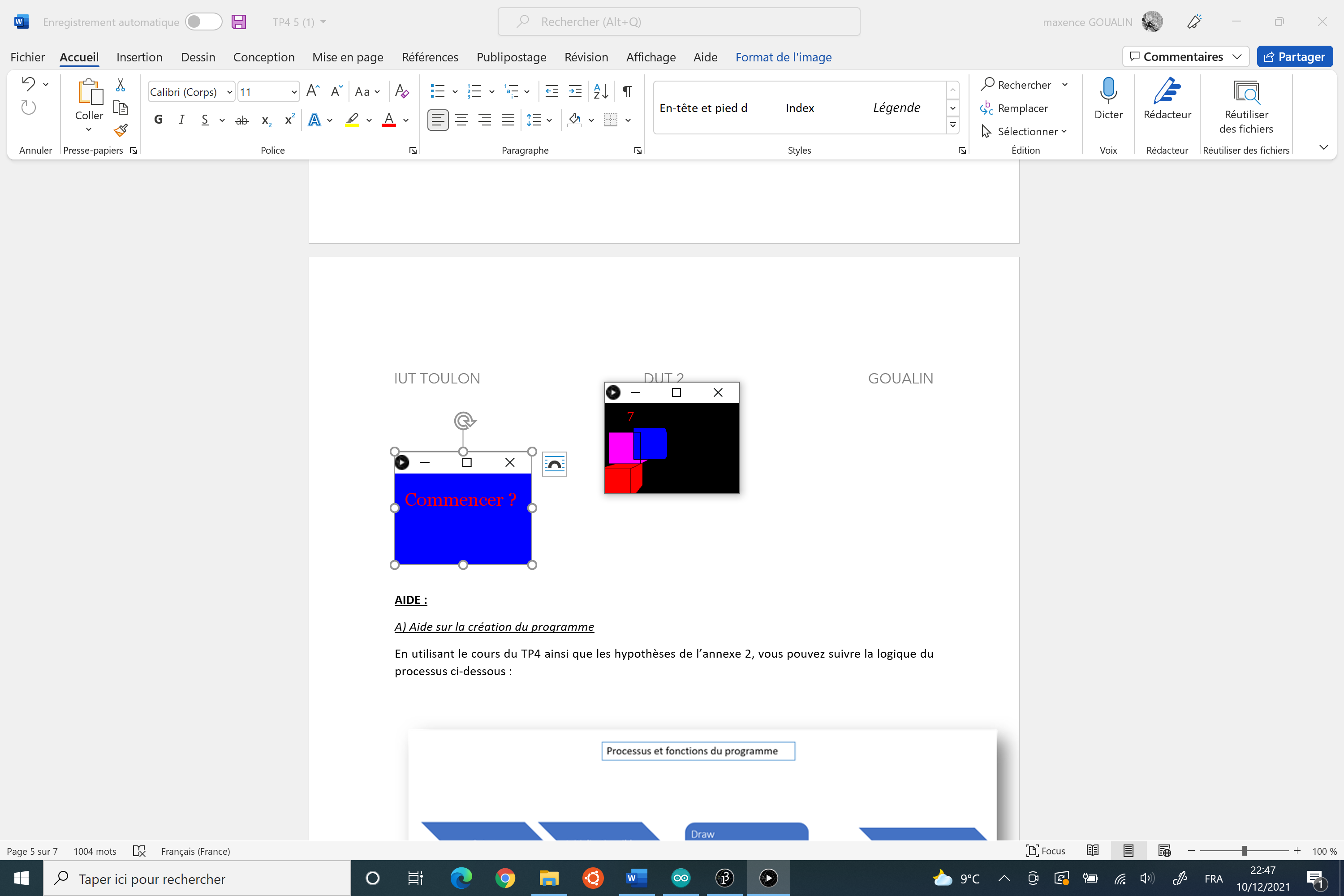
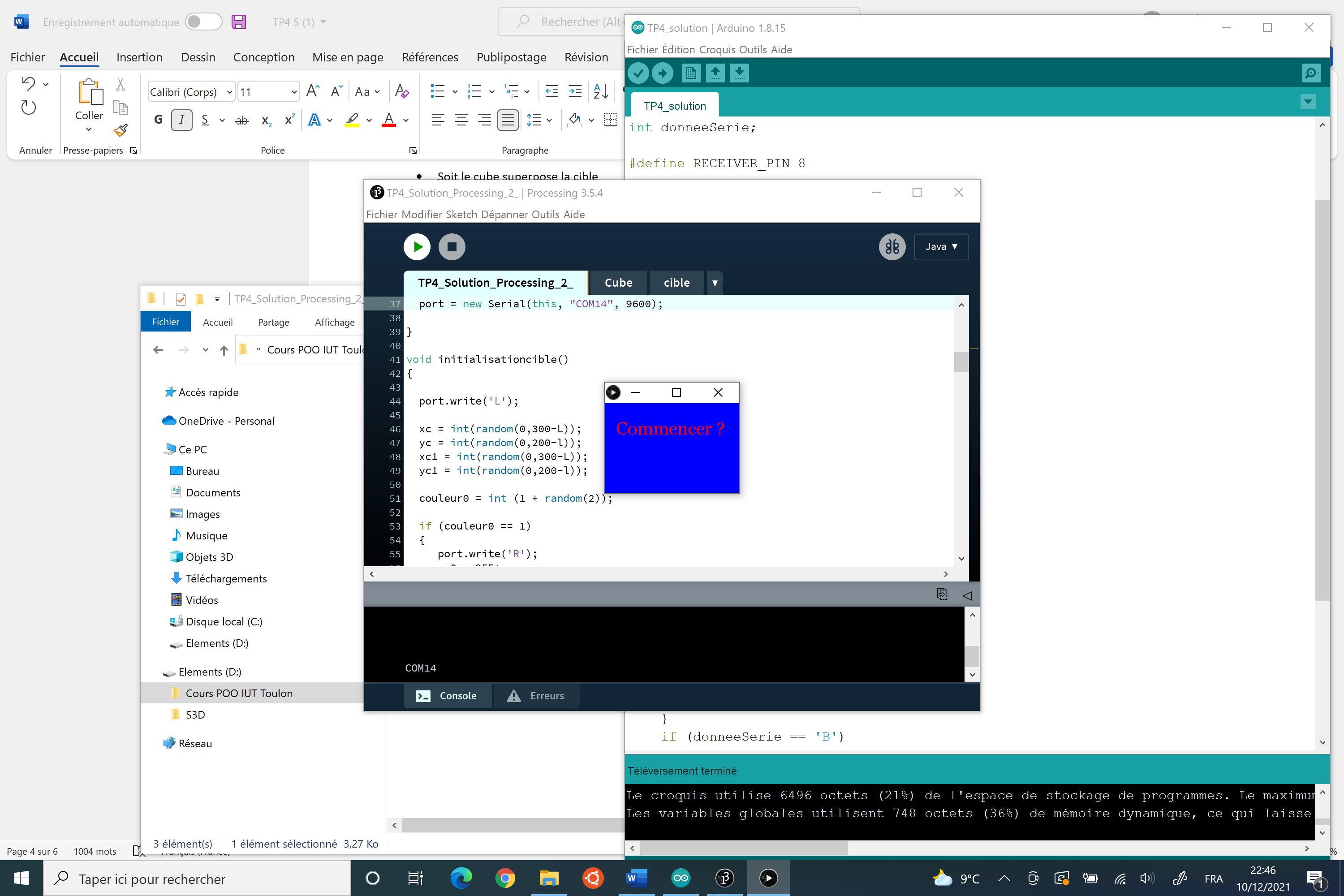
*.*

Modifier le code Arduino/Processing de la partie 1 afin de **(sur 5 points) :**

* Permettre d’allumer une LED de couleur aléatoire entre le rouge et le bleu.

Modifier le code Processing de la partie 1 afin de :

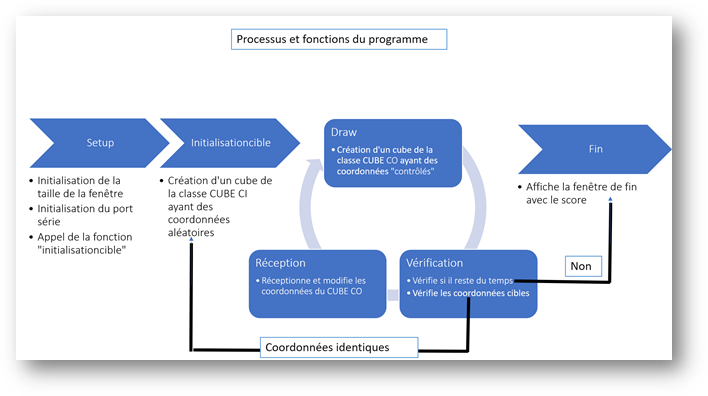
* Créer deux cubes cibles (ayant la même classe) de couleurs différentes (Rouge et Bleu) et de position aléatoire. **(sur 2 points)**
* Création d’une fenêtre de début de jeu avec le texte « Commencer ? » en utilisant le classe « PFont » **(sur 1 point)**
* Le jeu démarre après un clic de la souris et le timer commence à ce moment-là **(sur 2 points)**



**AIDE :**

*A) Aide sur la création du programme*

En utilisant le cours du TP4 ainsi que les hypothèses de l’annexe 2, vous pouvez suivre la logique du processus ci-dessous :



*B) Aide sur les fonctions*

* La fonction **Initialisationcible** utilise la fonction **random** :

**Random(Premier intervalle, deuxième intervalle)** retourne un “float” compris entre le premier intervalle et le deuxième intervalle.

Pour avoir un int, utiliser cette expression : **int(Random(Premier intervalle, deuxième intervalle))**

**Le deuxième intervalle est non compris.**

* La fonction **vérification** utilise la fonction **millis**:

**Millis()** retourne le temps du programme en milliseconde.

Pour avoir des secondes, diviser par 1000.

* La fonction **réception** compare deux chaînes de caractères :
  + Première chaîne : le code reçu par le port série en utilisant la fonction **vue dans le cours,**
  + Deuxième chaîne : le code mémorisé au §1.B.

Pour comparer les deux codes, utiliser la fonction : **match(code reçu, code mémorisé) ;** cette fonction retourne un tableau de String si les codes sont identiques, sinon retourne NULL.

* La fonction **fin** utilise la fonction **Stop()**: Stop() arrête le programme.

Annexe I : PIN Arduino NANO

Annexe II : Hypothèses de la partie A

**Tailles des cubes :** Longueur = 50 ; largeur = 50 ; épaisseur = 50 ;

**Position initiale du Cube CO** : X0 = 125 ; y0 = 75 (au milieu de la fenêtre 300\*200)

**Pas de déplacement du Cube CO** = 1

**Temps de la partie :** 60 secondes

**Gain de temps à chaque superposition** : 10 secondes