Tutorial Arduino

Semaine 1

Auteurs :
Bassiro TRAORE

26 janvier 2020

Table des matières

1 Simulateur

Set up

Créer un compte gratuitement sur https ://www.tinkercad.com.
Inviter une personne à votre projet

Dans notre cas , nous allons utiliser le lien ci-dessous
`https://www.tinkercad.com/things/imxlJxV5Vgm-super-rottis/editel?sharecode=WsEeAcEpz5KE16KIQpN0iC2qauqP8BMY`

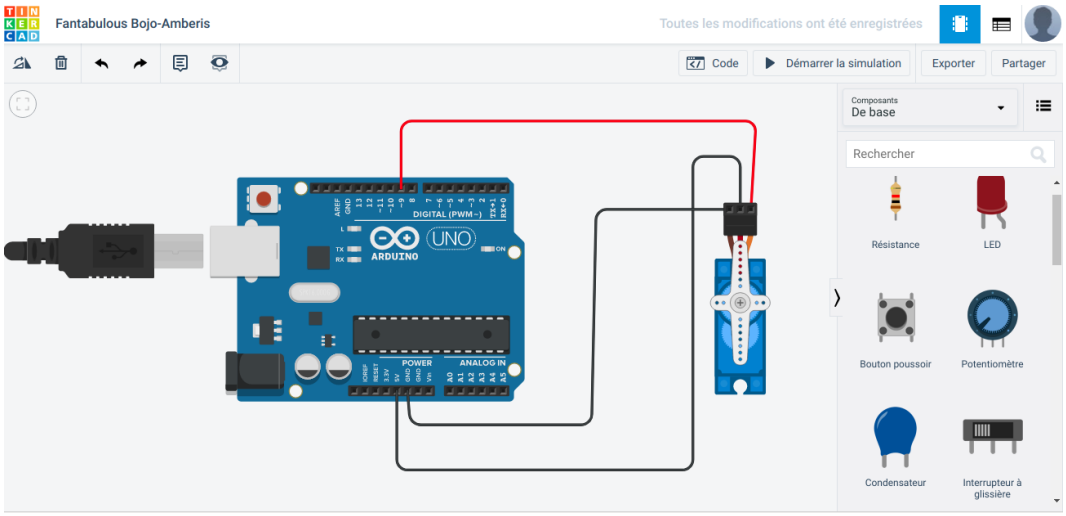

FIGURE 1 – Image du Simulateur online

La session Code permet de tester votre script , vous pouvez aussi ajouter des composants ainsi que des plaquettes .
Cependant , merci de bien-vouloir faire des captures et des sauvegardes afin d'avoir une copie de votre travail dans votre ordinateur.

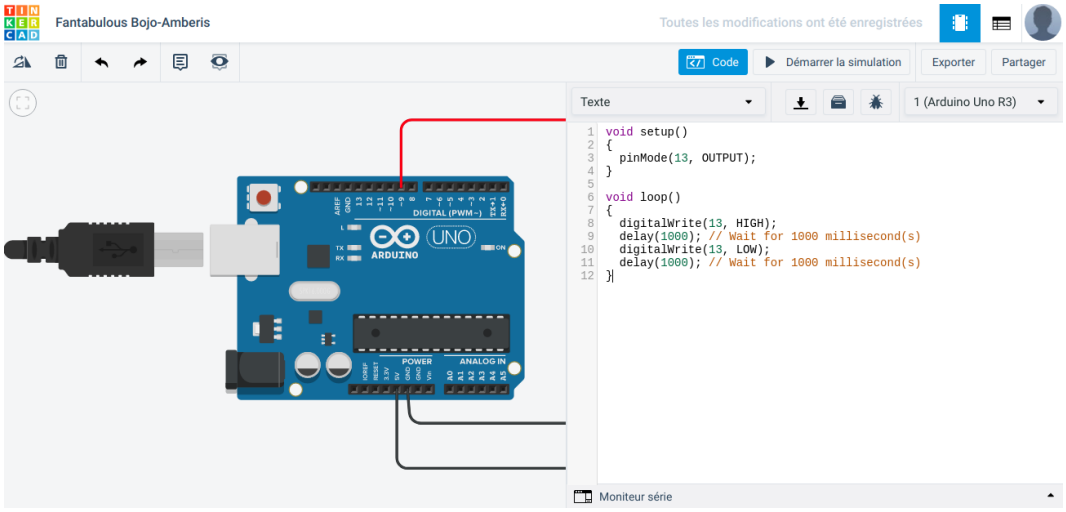

FIGURE 2 – Image du Simulateur avec l'option code

2 Programmation d'une fonction

Void

La creation d'une fonction est initier par le mot clé Void et le nom
Synthaxe : void name(parame)

exemple

Listing 1 – utilisation de Serial

```
void setup() {
} /* obligatoire avec Arduino */
void loop() {/* vide  */ } /* obligatoire avec Arduino */

void test(int param){
 } /* notre nouvelle fonction */
```

les types de données et leurs declarations

3 Programmation d'une Classe pour creer sa propre bibliothèque

FIGURE 3 – Image du Simulateur avec l'option code

Une classe est un element important de la programmation en orienté object.

Une classe possède des méthodes qui correspondent a des fonctions qui s'appliquent sur les objets.

lors de la creation d'une classe, il faut faire appel a un Constructeur .

le constructeur est une sorte d'initialison qui est faite dans la classe ainsi lorsque la classe est appeler le constructeur s'applique directement

Listing 2 – creation de classe

```
class Testons{
public :
Testons(); //Constructeur
// ...
private :
// ...
};
```

ses methodes peuvent être public ou prive.

méthode public et méthode private

Une méthode est une fonction qui permet d'effectuer une tache en programmation oriénté Objet.

une méthode est dite public si elle est accessible depuis l'exterieur. à la difference d'une méthode prive (private) inacessible depuis l'exterieur.

les attributs public et private sont valables aussi pour les attributs
exemple

Listing 3 – creation de classe

```
class Testons{
public :
Testons(int,int,int,int); //Constructeur
// ...
void test1(); // Méthode pour l'objet testons
private :
  int Test1; // variable 1
  int Test2; // variable 2
  int Test3; // variable 3
  int Test4; // variable 4
  int Test5; // variable 5
};
```

enfin nous avons comment construire une classe .
mais en C/C++ cela ne suffit pas .

pour pouvoir initialiser la classe il faut d'abord un fichier entête et un fichier programme.
les extensions de ses deux fichiers sont . h et . cpp.
par la suite ils sont installer dans arduino dans le repertoire library $\_1.x.yibraries$

Soit une declaration de fichier .h

Listing 4 – necessaire pour un fichier .h

```
#ifndef Testons_h
#define Testons_h
..
#endif
```

exemple concret avec notre fichier Testons

Listing 5 – necessaire pour un fichier .h

```
#ifndef Testons_h
#define Testons_h

#if ARDUINO < 100
#include <WProgram.h>
#else
#include <Arduino.h >
#endif

class Testons{
public :
Testons(int,int,int,int); //Constructeur
// ...
void test1(); // Méthode pour l'objet testons
private :
  int Test1; // variable 1
  int Test2; // variable 2
  int Test3; // variable 3
  int Test4; // variable 4
  int Test5; // variable 5
};

#endif
```

Par la suite après avoir creer notre fichier .h creons notre fichier .cpp
il s'agit du fichier mettant en place la classe construite

Listing 6 – necessaire pour un fichier .h

```
#include "Testons.h"
```

```
#define WAITTIME 20

//Constructeur parametre
Testons::Testons( int A, int B, int C, int D){
TestA = A;
TestB = B;
Teste = C;
TestD = D;
pinMode (TestA, OUTPUT) ;
pinMode (TestB, OUTPUT);
pinMode (TestC, OUTPUT);
pinMode (TestD, OUTPUT);
}

//Methode pour lancer le de
void Testons:: Test(){

int number = random(1, 7);
digitalWrite (TestA, number%2 ! = o?HIGH :LOW);
digital Write (TestB, number>1 ?HIGH : LOW);
digitalWrite (TestC, number>3?HIGH :LOW);
digitalWri te ( TestD, number==6 ?HIGH : LOW);
delay(WAITIIME); //Ajouter une courte pause
}
```

4 initiation au code en STL

Serial

Serial (ou comment lire et écrire)

La librairie Serial est une librairie essentielle du langage Arduino qui permet de visualiser sur le PC des messages reçus depuis la carte Arduino ou de commander la carte Arduino.

En couplant l'utilisation de cette librairie avec l'interface programmable graphique Processing côté PC, on dispose d'un outil extrêmement puissant pour réaliser toute sortes d'affichages graphiques sur le PC ou d'interactions entre la carte et le PC (commande de la carte Arduino avec la souris ou le clavier !).

Listing 7 – utilisation de Serial

```
 int x=0;
void setup() {
  /*Inicializa a comunicao serial a uma
  taxa de 9600 baunds por segundo*/
  Serial.begin(9600);
}

void loop() {
  /*Serial.print("Contando: ");    //Imprime a palavra "Contando"
  Serial.print(x);                //Imprime o valor da variavel x
  Serial.println(" segundos."); //Imprime a palavra " segundos."*/
  Serial.println("Contando: "+(String)x+" segundos.");
  delay(1000);                    //Espera 1 segundo
  x++;                            //Soma x+1
}
```

Les fonctions de la bibliothèques Serial

les fonctions sont :

begin()
available()
read()

flush()
print()
println()
write()

Serial.begin()

Cette fonction doit être appelé au moins une fois generalement dans la fonction Setup() du programme , afin de définir la vitesse utilisée sur la liaison serié . la synthaxe est la suivante

Listing 8 – initialisation de Serial

```
Serial.begin(vitesse);
```

Les vitesses peuvent être 300,1200,2400,4800,900,19200,14400,19200,28800,38400 etc

Serial.print()

Affiche les données sur le port série sous forme lisible pour les humains (texte ASCII). Cette instruction peut prendre plusieurs formes.
Les nombres entiers sont affichés en utilisant les caractères ASCII pour chaque chiffre.
Les nombres à virgules (float) sont affichés de la même façon sous forme de caractères ASCII pour chaque chiffre, par défaut avec 2 décimales derrière la virgule.
Les valeurs de type byte sont affichés sous la forme d'un caractère ASCII.
Les caractères et les chaînes sont affichés tels quels.

Serial.print(val) ou Serial.print(val, format)

Exemple daffichage

Serial.print(byte(78)) ; // affiche "N" (dont la valeur ASCII est 78)
Serial.print('N') ; // affiche "N"
Serial.print("Hello world.") ; // affiche "Hello world."

Serial.print(78, HEX) ; // affiche "4E"
Serial.print(1.23456, 0) ; // affiche "1"
Serial.print(1.23456, 2) ; // affiche "1.23"
Serial.print(1.23456, 4) ; // affiche "1.2346"

Serial.println()

Description
Affiche les données sur le port série suivi d'un caractère de "retour de chariot" (ASCII 13, or ”̊) et un caractère de "nouvelle ligne" (ASCII 10, or ”). Cette instruction a par ailleurs la même forme que l'instruction Serial.print()

Autres fonctions de la bibliothèques Serial

if (Serial)
end()
find()
findUntil()
parseFloat()
parseInt()
peek()

readBytes()
readBytesUntil()
setTimeout()
serialEvent()

5 Biblothèque pour la carte SD

Cette bibliothèque ouvre des possiblites de stockage des de données impressionnantes puisques les cartes SD disposent de 4 à 8 Go despaces
pour initialiser cette biblithèque , il faut faire
include <SD.h> La communication entre la carte Arduino et la carte mémoire SD utilise la communication SPI, laquelle utilise les broches 11, 12 et 13 via une carte un shield Ethernet . Cette meme carte permet dutiliser les options internet en mode DHCP sur cette carte ardunio

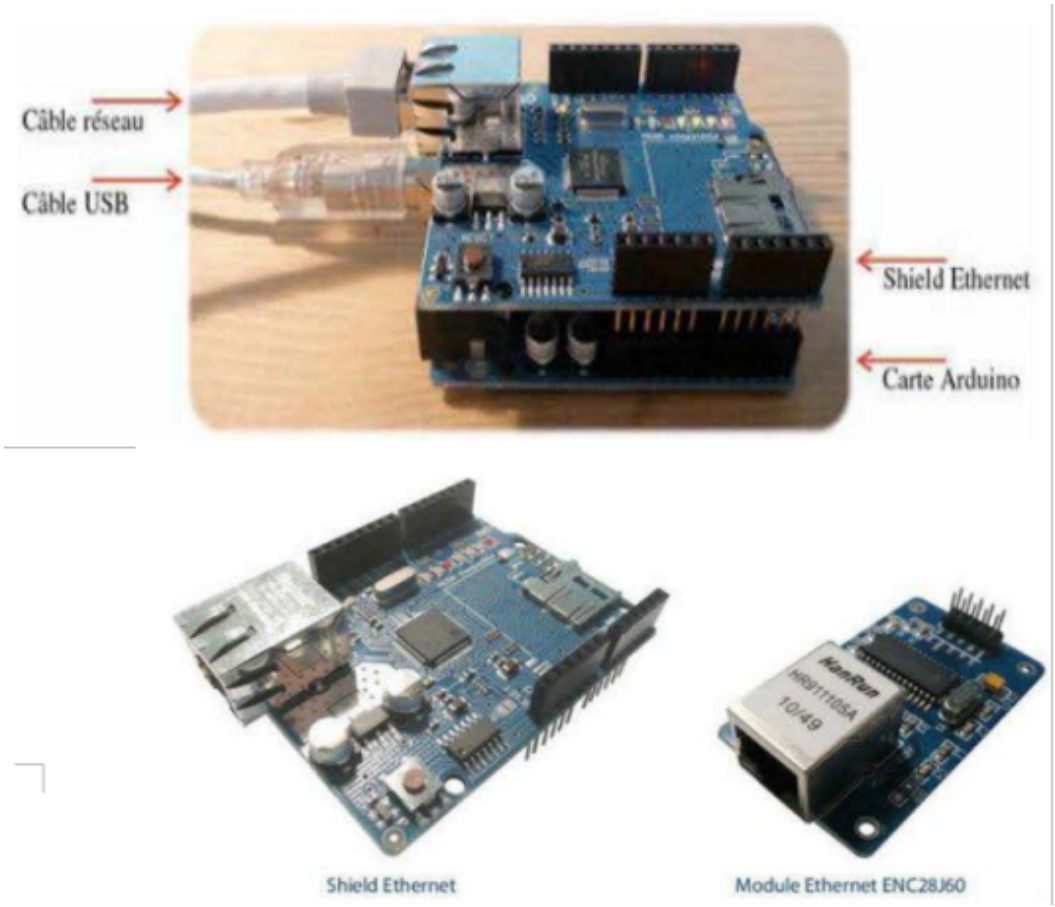

FIGURE 4 – Image du Simulateur avec l'option code

les fonctions de cette bibliothèque

begin()
exists()
mkdir()
open()
remove()
rmdir()

available()
close()
flush()
peek()
position()
print()
println()
seek()
size()
read()
write()
isDirectory()
openNextFile()
rewindDirectory()

SD.begin()

Initialise la librairie SD et la carte SD. Cela lance l'utilisation du bus SPI (broches numériques 11,12 et 13 sur la plupart des cartes Arduino) et initialise la broche de sélection du circuit intégré (chip select), qui est par défaut la broche matérielle SS (broche 10 sur la plupart des cartes Arduino).

Listing 9 – utilisation de Serial

```
#include <SD.h>
 void setup()
{
 Serial.begin(115200); // utiliser le meme debit coté Terminal Serie
 Serial.println("Initialisation de la SD card...");
 pinMode(10, OUTPUT); // laisser la broche SS en sortie - obligatoire avec librairie SD
if (!SD.begin(10)) { // si initialisation avec broche 10 en tant que CS n'est pas réussie
 Serial.println("Echec initialisation!"); // message port Série
 return; // sort de setup()
}
 Serial.println("Initialisation reussie !"); // message port Série
}
void loop(){
}
```

SD.exist()

Cette fonction teste si un fichier ou un répertoire existe sur la carte mémoire SD.
Synthaxeă: SD.exists(filename) et renvoi true ou false

SD.mkdir()

Cette fonction crée un répertoire sur la carte mémoire SD.
Cette fonction crée également les répertoires intermédiaires qui n'existe pas encore ; par exemple : SD.mkdir("a/b/c") ; rla Valeur renvoyée est True ou False.

SD.rmdir()

Cette fonction efface un répertoire (remove dir) de la carte mémoire SD. Le répertoire doit être vide.
Synthaxeă: SD.rmdir(filepath),
les valeurs renvoyéesă
true : si le répertoire a bien été effacé

false : si le répertoire n'a pas pu être effacé.
Si le répertoire n'existe pas, la valeur renvoyée n'est pas spécifiée.

SD.remove()

Cette fonction efface un fichier de la carte mémoire SD.
Sythanxeă: SD.remove(filename)
true : si le répertoire a bien été effacé
false : si le répertoire n'a pas pu être effacé.
Si le répertoire n'existe pas, la valeur renvoyée n'est pas spécifiée.

Ouverture dun fichier sur la carte memoire On déclare une variable de type fichier : File nomFile ; On ouvre un fichier ainsi : File f = SD.open("data.txt", FILE$_{WRITE}$);

file.close()

Cette fonction ferme le fichier, et s'assure que toute donnée écrite dans ce fichier est physiquement enregistrée sur la carte mémoire SD.

file.read()

Cette fonction lit un fichier.

file.peek()

Cette fonction lit un octet dans un fichier sans avancer au suivant.lappel successive correspond ainsi a la fonction read initialement vue.

file.write()

Cette fonction écrit des données dans un fichier.
Syntaxeă: file.write(data) ou file.write(buf, len).
data : l'octet(byte), le caractère(char) ou la chaîne de caractère (char *) à écrire dans le fichier

file.print()

Cette fonction sert a afficher le contenu du fichier lu dans le module SD .
La synthaxeă:
file.print(data)
file.print(data, BASE)

file.println()

cette fonction est quasi similaire a la fonction fil.print() à la difference prêt que à chaque ligne il y a un retour chariot .

file.flush()

Cette fonction s'assure que les données écrites dans un fichier ont été physiquement enregistrées sur la carte mémoire SD.
Ceci est fait automatiquement également lorsque le fichier est fermé avec la fonction close()

file.size()

La taille du fichier lu en nombre d'octets , les conversions par la suite en Mega octets , Mega bits ,etc , référerez vous a votre cours sur la conversion.

faire attention ici le type de la donnée retourne unsigned long elle necessitera une declaration initiale lors de la programmation en C ou C pp

file.available()

Cette fonction vérifie si des octets sont disponibles en lecture dans le fichier.
Elle retourne un int

file.position()

Cette fonction renvoie la position courante à l'intérieur du fichier, c'est-à-dire la position à laquelle le prochain octet sera lu ou écrit).
Elle retoourne un la valeur de la position à l'intérieur du fichier type long

file.seek()

Cette fonction permet de se placer à une nouvelle position, qui doit être comprise entre 0 et la taille du fichier (inclusif).
Sythnaxeă: file.seek(pos)
file : une instance de l'objet File (renvoyée par la fonction SD.open())
pos : la position à laquelle se placer type unsigned long
Valeur renvoyéeă: true : en cas de réussite, false : en cas d'échec.

file.isDirectory()

Les répertoires (ou dossiers) sont une sorte particulière de fichiers : cette fonction permet de savoir si le fichier courant est un répertoire ou non.
Syntaxeă: file.isDirectory()
Valeur retournéeă: Renvoie un boolean(true si le fichier courant est un répertoire, false sinon.)

file.openNextFile()

Renvoie le fichier ou dossier suivant dans un répertoire.
Syntaxeă: file.openNextFile()
la valeur retournéeă: char (le nom du fichier ou répertoire suivant.)

file.rewindDirectory()

Se place au niveau du premier fichier dans un répertoire. Utilisée en association avec openNextFile() Synthaxeă: file.rewindDirectory()

Listing 10 – utilisation de Serial

```
#include <SD.h>
File root;
void setup()
{
Serial.begin(9600);
pinMode(10, OUTPUT);
SD.begin(10);
root = SD.open("/");
printDirectory(root, 0);
Serial.println("done!");
}

void loop()
{
// nothing happens after setup finishes.
}
void printDirectory(File dir, int numTabs) {
while(true) {
File entry = dir.openNextFile();
if (! entry) {
// si pas de nouveau fichier
// renvoie le premier fichier dans le répertoire
dir.rewindDirectory();
break;
}
for (uint8_t i=0; i<numTabs; i++) {
Serial.print('\t');
}
Serial.print(entry.name());
if (entry.isDirectory()) {
Serial.println("/");
printDirectory(entry, numTabs+1);
} else {
// les fichiers ont une taille, pas les répertoires
Serial.print("\t\t");
Serial.println(entry.size(), DEC);
}
}
}
```

6 insertion d'un LCD

l'ecran nous permettra d'afficher à l'utilisateur un témoin de la valeur de la température ainsi que l'état du sorte .

6.1 code

Listing 11 – initialisation de Serial

```
const int sensorPin = A0;
int reading;
float voltage;
float temperatureC;
```

```
void setup()
{
 Serial.begin(9600);
}

void loop()
{
reading = analogRead(sensorPin);
voltage = reading * 5.0/1024;
Serial.print (voltage);
Serial.println(" volts");
temperatureC = (voltage - 0.5) * 100 ;
Serial.println("Temperature is: ");
Serial.print(temperatureC);
Serial.println(" degrees C");
delay(1000);
}
```

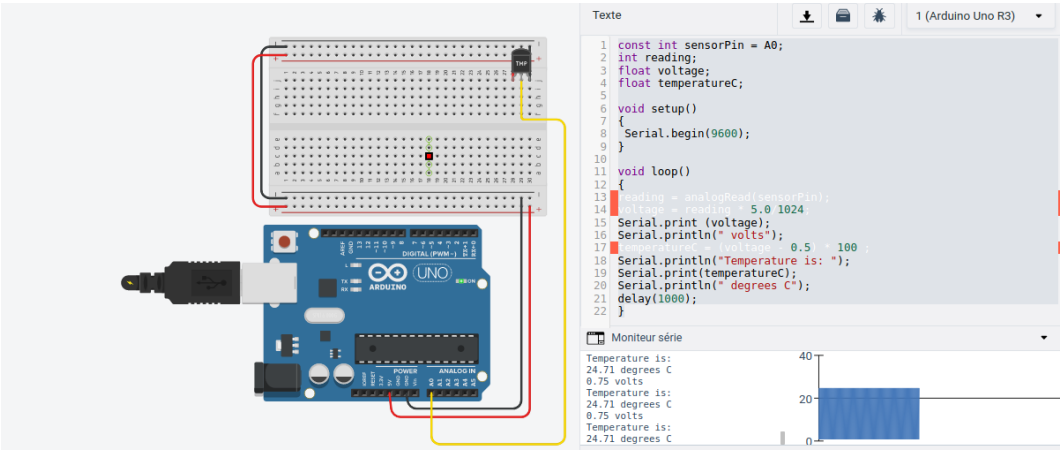

FIGURE 5 – Image du Simulateur online

Listing 12 – initialisation de Serial

```
#include <Servo.h>
Servo myservo;//call Servo
int pos = 0; // initialisation de la position

const int sensorPin = A0;
int reading;
float voltage;
float temperatureC;

void setup()
{
 Serial.begin(9600);
  myservo.attach(9);// indcation du port de lecture
}

void loop()
{
reading = analogRead(sensorPin);
voltage = reading * 5.0/1024;
Serial.print (voltage);
Serial.println(" volts");
temperatureC = (voltage - 0.5) * 100 ;
Serial.println("Temperature is: ");
Serial.print(temperatureC);
Serial.println(" degrees C");
```

```
temperatureC= 27;
  if ( temperatureC<=20){
  Serial.println(pos);
    for(pos = 0; pos < 180; pos += 1)
    {
    myservo.write(pos);

    }
  }
  if ( temperatureC>20){

Serial.println(pos);
    for(pos = 0; pos>=180; pos+=1)
    {
    myservo.write(pos);
      delay(900);
    }
  }
/*for(pos = 0; pos < 180; pos += 1)
{
myservo.write(pos);
delay(15);
}
for(pos = 180; pos>=1; pos-=1)
{
myservo.write(pos);
delay(15);
}
*/

delay(1000);
}
```

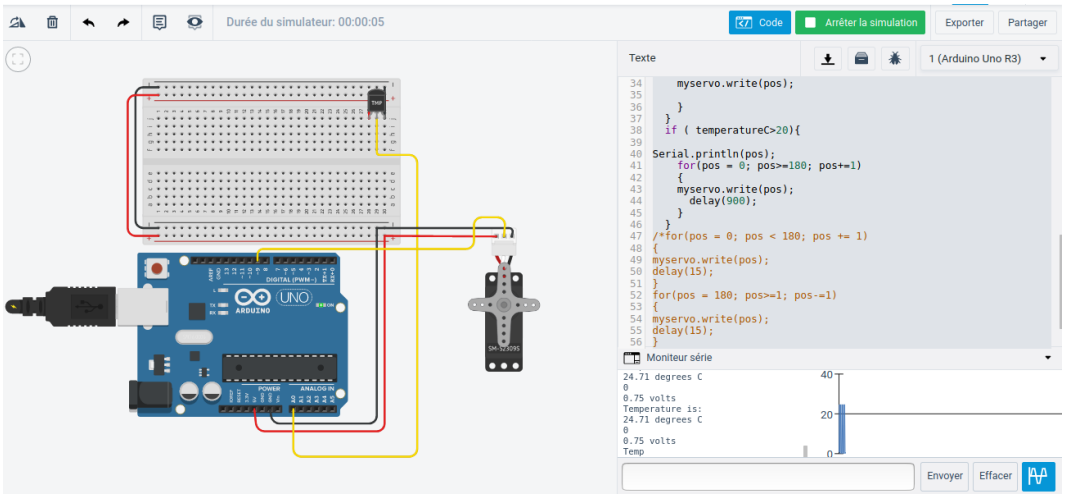

FIGURE 6 – Image du Simulateur online