

# BE C++: Robot planteur de graines

**Programmation orientée objet**

Département Génie Electrique et Informatique  
INSA Toulouse  
11 Mai, 2022

Mazin Zaouali  
Clément Sevillano

# 1 Introduction

Le but de ce bureau d'études est de développer un objet contrôlé par une carte EST8266/Arduno, combinant actionneurs et capteurs. Ce prototype devait respecter une programmation orientée objet. Ainsi, nous avons décidé de développer un "Robot planteur de graines". Pour résumer, le robot va planter des graines en effectuant plusieurs lignes de plantations comme représentées ci-dessous.

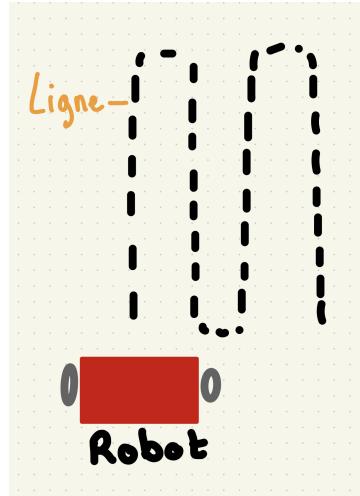


Figure 1: Représentation déplacement du robot.

L'utilisateur a donc le choix du nombre de lignes que le robot doit effectuer avec une limite à 8 lignes. En plus de cela, l'utilisateur a la possibilité de choisir entre deux types de graines pour chaque ligne : les roses ou les tulipes. Puis, il pourra à la fin de la plantation voir le nombre de graines qui ont été plantées. Concernant le fonctionnement, le robot avance à l'aide de deux grandes roues contrôlées par des "stepper motor" et une roue de stabilisation. Afin de planter les graines, nous avons utilisé un servo-moteur. Pour choisir le type de graines et le nombre de lignes, nous avons pris une matrice de LED pour l'afficher et un joystick pour le choix et le lancement du robot. Pour finir, un écran OLED est présent pour afficher le nombre de graines plantées après chaque ligne.

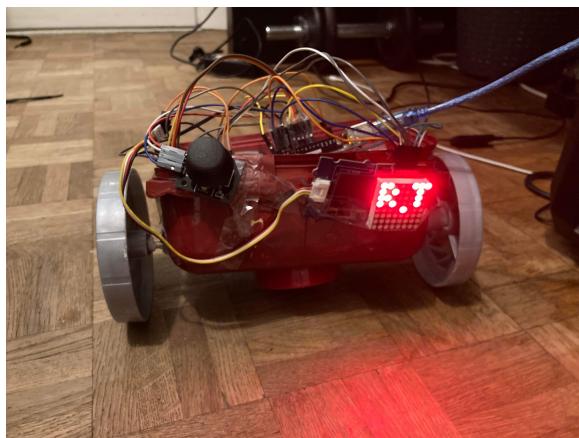
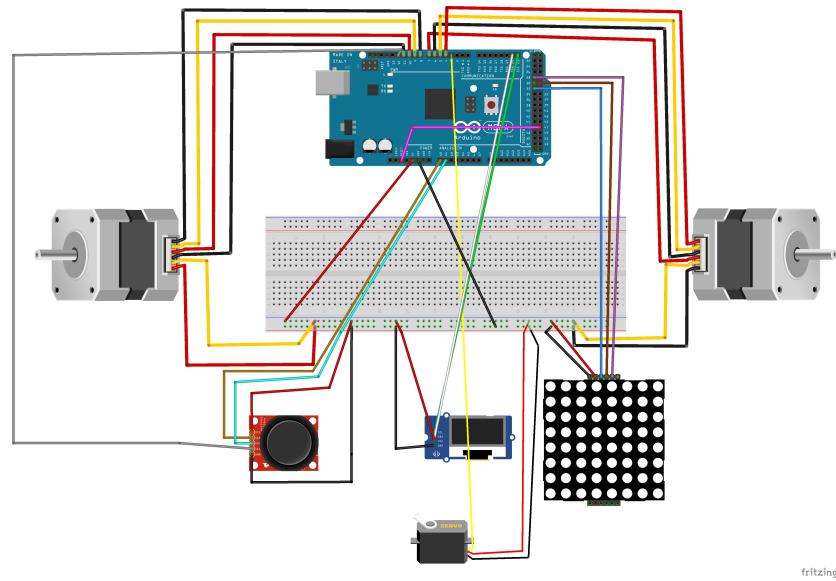


Figure 2: Robot vue de face.



Figure 3: Robot vue du haut.

## 2 Schéma fonctionnel du robot



fritzing

Figure 4: Schéma fonctionnel avec tous les composants du robot.

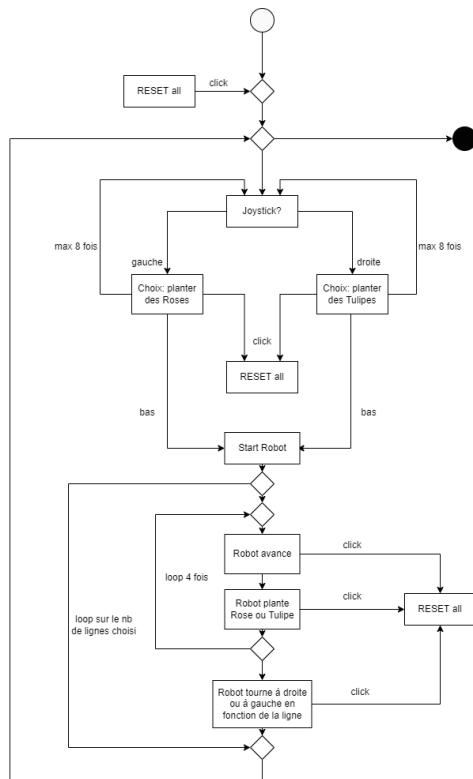


Figure 5: Schéma de fonctionnement logiciel

### 3 Diagramme de classe du code de notre robot

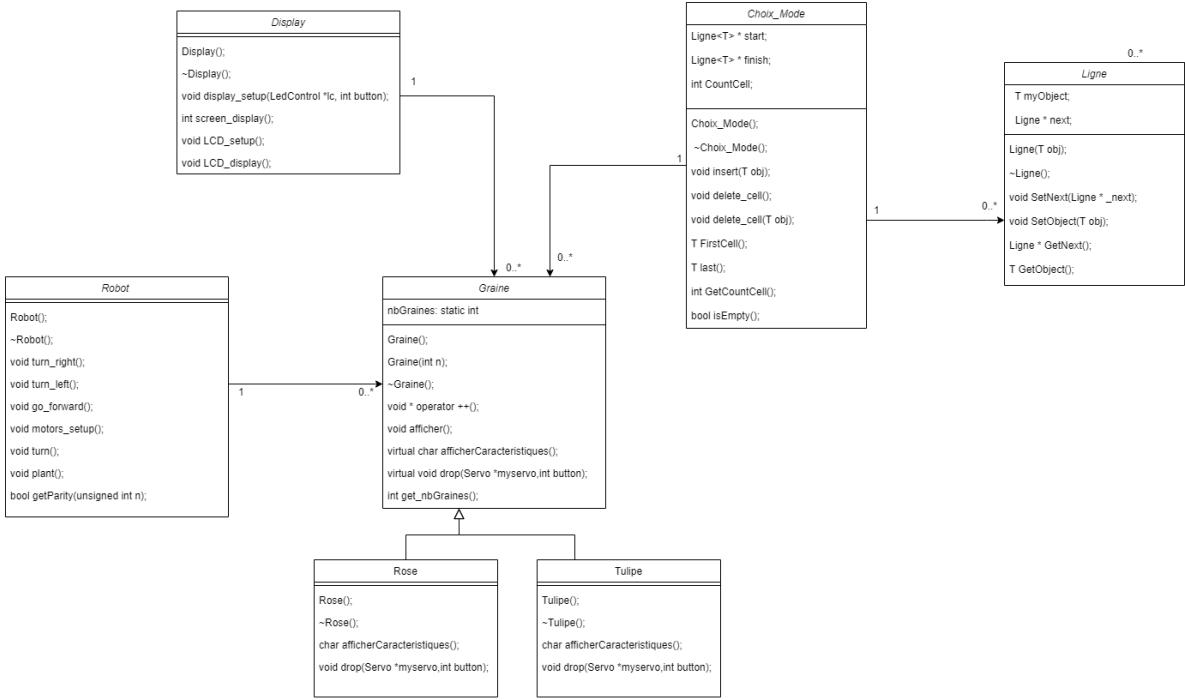


Figure 6: Diagramme de classe.

### 4 Conclusion

Ce projet nous a posé de nombreux problèmes. En effet, lorsque nous avons eu l'idée du robot planteur de graine, nous avions pensé au code qui était réalisable. Cependant, nous n'avions pas réfléchi sur les problèmes matériels. En effet, des erreurs de contact, une perte de courant, un nombre de câbles limités, un robot construit avec des pièces de différents objets. Tout ceci est une vraie source de problème car assemblé, le code peut compiler, mais le robot ne marche pas, le problème vient donc d'autres choses que de l'algorithmie. De plus, nous avons décidé d'utiliser une Arduino Mega 2560 au lieu de l'ESP8266. L'Arduino Mega, grâce à ses nombreux ports, est plus optimale pour les branchements des différents composants. Cependant, les cartes Arduino ne permettent pas de coder des exceptions, le code ne compile pas contrairement à l'ESP8266. Nous avons donc du coder une exception avec l'ESP pour ensuite l'intégrer en commentaire dans notre programme Arduino pour montrer que nous l'avions fait. Concernant la programmation, la construction en C++ permet une bonne structuration des classes, des méthodes et des attributs. Il est alors facile de trouver l'erreur en cas de problèmes. De plus, la surcharge de l'opérateur "`++`" nous a permis de compter le nombre d'objet "Graine" planté ce qui nous a permis de l'afficher directement sur l'écran LCD. L'utilisation de "virtual" entre classes filles et classe mère nous a permis d'identifier facilement s'il s'agissait d'une Rose ou d'une Tulipe lorsqu'on faisait du debugging manuellement (en utilisant des `Serial.print()`) et en utilisant un pointeur sur un objet de la classe mère Graine. Enfin, nous avons décidé de créer notre propre liste chaînée en utilisant les templates, au lieu d'utiliser STL, et cela pour qu'on puisse d'une part bien comprendre le fonctionnement des listes chaînées et de nous familiariser avec la syntaxe des templates, et d'autre part pour avoir des classes qui nous paraissaient plus claires et plus "parlantes" en termes de compréhension (Liste-> Choix Mode, Cellule-> Ligne). Nous avons aussi eu certains soucis notamment avec la compréhension des fichiers "tpp.", mais nous avons su chercher sur les différents forums afin de trouver nos réponses à nos problèmes.