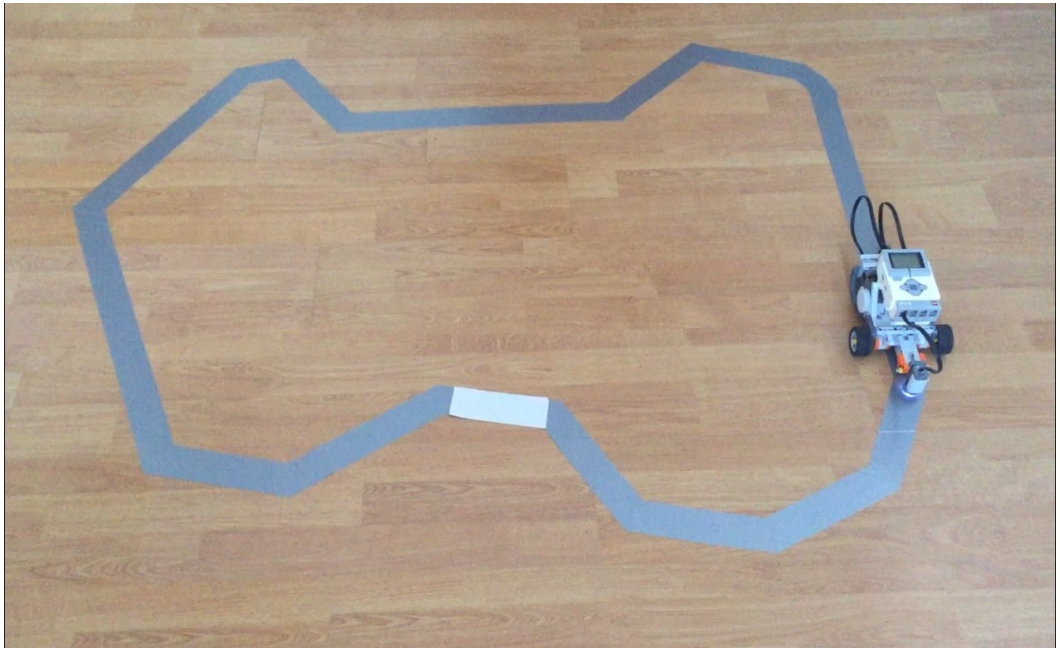
Projet Mindstorm

Benyahia Meriem, Montigne Philippe, Florian Szczepanski

Introduction

Suivre une ligne



Fonctionnalités

- Apprentissage de plusieurs couleurs
- Suivi de ligne droite et courbe
- Exécute différentes fonctions en scannant la couleur associé
  - Demi-tour
  - Arrêt

Fonctionnalités

- **Apprentissage d’un circuit pendant un parcours**
- **Peut avancer sans ligne avec des fonctions :**
  - Pour tourner d’un certain angle
  - Pour avancer d’une certaine distance
- **Calcul de points « intelligents » pour parcourir le circuit plus rapidement à l’aveugle**

Organisation et Architecture

- **Décomposition du problèmes**
  - Reconnaissance des couleurs
  - Suivi de ligne
  - Mémorisation du circuit
- **2 projets Eclipse**
  - ColorGestion : apprentissage des couleurs
  - Line Follow : FollowLine pour suivre une ligne et réagir à d'autres couleurs / FollowMem pour suivre une ligne et reproduire le circuit à l'aveugle

Conception

- **Gestion de la couleurs**
  - Apprentissage des couleurs au robot
- **Suivi de ligne**
  - Premier algo « naïf » → problème de virage serré
  - Deuxième algo pour résoudre ce problème
  - Tentative de troisième pour améliorer la vitesse
- **Mémorisation de la ligne**
  - Utilisation du tachymètre pour apprendre la ligne

Difficultés rencontrés

- **Problème de connectique de boîtier**
  - Tous les tests ont été fait avec le mac (qui était le seul à réussir à se connecter au boîtier)
- **Capteur défectueux**
  - Changement de capteur
- **Altération de la reconnaissance des couleurs sur un circuit bombé**
  - Réajustement de la position du robot

Programmation

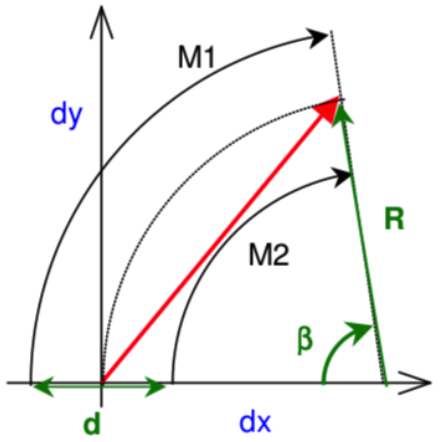
```java
public class MoveMem {
    private float distanceA;
    private float distanceD;
    private float angle;
    private float corvatureRadius;
    private float dx;
    private float dy;

    private static float DIA = 12.6f;

    public MoveMem(float dA, float dD){
        distanceA=dA;
        distanceD=dD;
    }

    public void createInfo(){
        corvatureRadius=18.9f;
        angle=((distanceA+distanceD)/2)/(corvatureRadius);

        dx =(float) (corvatureRadius*(1-Math.cos(angle)));
        if(distanceA > distanceD){
            dx=-dx;
        }
        dy= (float) (corvatureRadius*Math.sin(angle));
        if(distanceD>distanceA){
            angle=-angle;
        }
    }
```



d : Distance entre les roues

Mi : Distance parcourue par une roue

β : variation d'angle décrit par le robot

R : Rayon de courbure

$M_1=\beta.(R+\frac{d}{2})$

$M_2=\beta.(R-\frac{d}{2})$

$R=\frac{M_1+M_2}{2.\beta}$ $\quad dx=R.(1-\cos(\beta))$

$\beta=\frac{M_1-M_2}{d}$ $\quad dy=R.\sin(\beta)$

Déroulement du GIT

- ≈ 40 commits
- 25 premier semestre / 15 second semestre
- Gros commit au lieu de plusieurs petits
- 1 seul pc fonctionnel

Conclusion

- **Difficultés**
  - Le robot ne fait pas partie du code
  - Gestion de la précision du robot
- **Pour la suite**
  - Circuit à embranchements
  - Suivi de ligne plus précis

Merci pour votre attention