Soutenance de Projet Long

Par :

- *Salim ABDELFETTAH*
- *Hassane GACI*
- *Mounir Halit*

Jury :

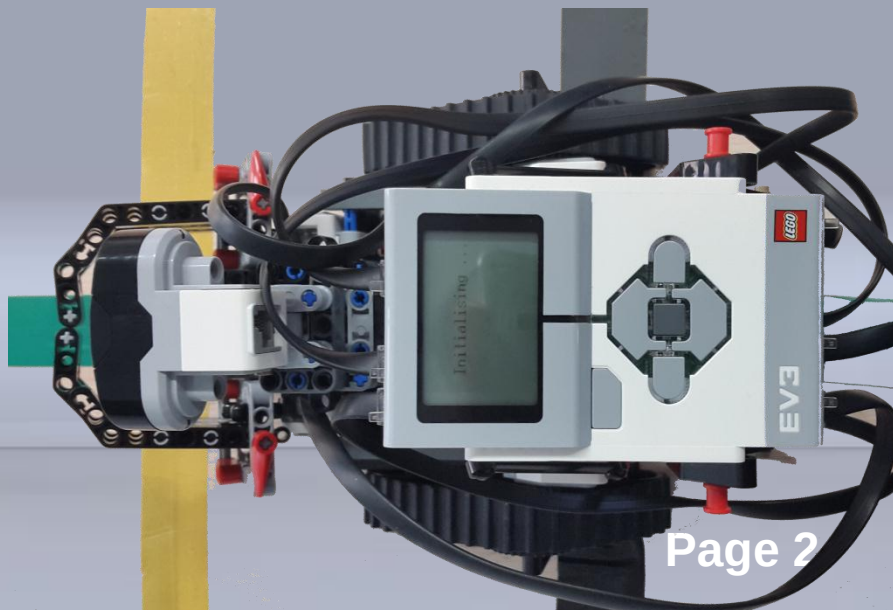
- *Yann Regis-Gianas*
- *Ines Klimann*
- *Jean-Baptiste Yunès*

université PARIS DIDEROT PARIS 7

Le problème

Le problème du projet consiste en la conception d'un programme autonome exécutable par une brique LEGO (Mindstorm EV3) pour la capture d'objets (le maximum possible).

Le programme une fois conçu et compilé servira dans une compétition où des robots LEGO s'affrontent deux par deux pour capturer des palettes.



Fonctionnalités du programme

Les fonctionnalités du programme sont comme suit :

- Détection des couleurs (deux capteurs)
- Calibrage des capteurs de couleurs
- Suivi de ligne (ajustement quand nécessaire)
- Détection des objets (devant le robot)
- Capture de l'objet
- Relâchement de l'objet

Architecture du programme

1ère architecture choisie – BDI : répudié pour les causes suivantes :

- **Code (Jason) non compilable directement au niveau de la brique (nécessité d'une connexion avec un ordinateur qui exécute le programme et lui transmettant les instructions à exécuter)**

- **Temps d'exécution et de réponse allongé (en raison de la communication entre la brique et l'ordinateur)**

Architecture du programme

2ème architecture choisie – Behaviour : gardé pour les causes suivantes :

- **Code compilable directement au niveau de la brique (déjà disponible sous LeJos)**
- **Décomposition du problème principale en sous problèmes sous forme de comportements (behaviours)**
- **Facile et intuitif**

Modules

Les modules sont :

- **Motors : offre les services liés aux moteurs ; avancer, reculer, tourner, capturer, relâcher, …**

  RegulatedMotors, HandMotor

- **Sensors : offre les services liés aux capteurs ; détection des couleurs/ objets/ obstacles, calibrage des capteurs, …**

  ColorSensors, IRSensor, TouchSensor

- **Behaviour : définit le comportement du robot (à partir des données reçus par les différents capteurs ; exécute une action effectué par les moteurs)**

Move, Adjust, Intersection, PreRelease, Release, PostRelease, SpecialAdjust, Catch

Difficultés rencontrés

Les difficultés rencontrés :

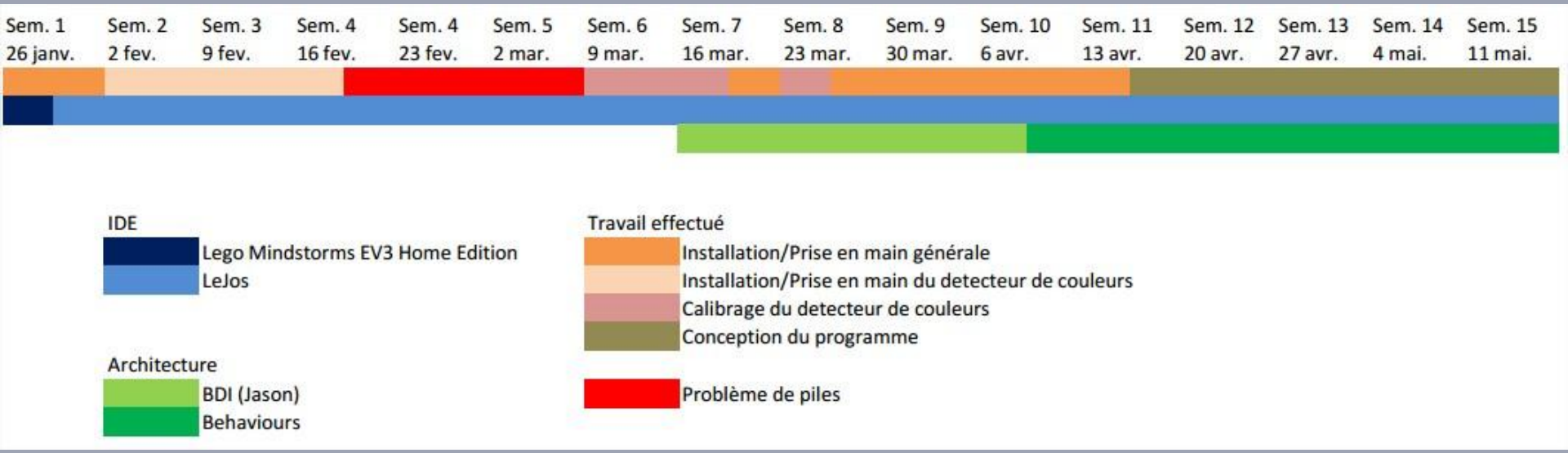
- **Algorithme de suivi de ligne non fonctionnel (du à l'emplacement des capteurs de couleurs)**
  → Solution : Montage des capteurs de couleurs à l'avant du robot

- **Inconsistance des résultats retournés par les capteurs de couleurs**
  → Solution : mise en place de la procédure de calibrage

- **Gestion des comportements du robot**
  → Solution : ajout de variables d'états/contrôles

- **Incohérence du comportement du robot**
  → Solution : tests et correctifs
  Partiellement réglé

Evolution du projet dans le temps



Code – behaviour : Moving

```
public boolean takeControl() {
        return !Controller.isCatchingOrReleasing()
        && ColorSensors.leftAndRightEquals(ColorSensors.WHITE);
}

public void action() {
        suppress = false;
        Controller.setOktoAdjust(true);
        RegulatedMotors.moveForward();
        while (!suppress) {
                Thread.yield();
        }
        RegulatedMotors.stopMoving();
}

public void suppress() {
        suppress = true;
}
```

Code – behaviour : Adjust

```
public boolean takeControl() {
        return Controller.isOkToAdjust() && ColorSensors.isColorsDifferent();
}

public void action() {
        Controller.setOktoAdjust(false);
        RegulatedMotors.stopMoving();
        int colorLeft = ColorSensors.getLeftColorId();
        int colorRight = ColorSensors.getRightColorId();
        if (colorLeft != ColorSensors.WHITE && colorRight !=
        ColorSensors.WHITE) {
                Controller.setOktoAdjust(false);
        } else if (colorLeft != ColorSensors.WHITE) {
                RegulatedMotors.turnleft();
                while (!ColorSensors.leftColorEqualsTo(ColorSensors.WHITE));
                RegulatedMotors.stopMoving();
        } else {
                RegulatedMotors.turnright();
                while(!ColorSensors.rightColorEqualsTo(ColorSensors.WHITE));
                RegulatedMotors.stopMoving();
        }
}
```

Code – action : Release

```
public static void releaseObject(int speed) {
        if (state == CATCHED) {
                hand.setSpeed(speed);
                hand.backward();
                while(!TouchSensor.isPressed());
                hand.stop(true);
                hand.forward();
                while(TouchSensor.isPressed());
                hand.stop(true);
                state = RELEASED;
        }
}
```

Conclusion

Ce projet nous a permit de voir les différents aspects et difficultés (liés au monde physique, inexactitude des résultats, …) **que l'on peut rencontrer dans la programmation robotique mais aussi dans la gestion d'un projet.**

Le comportement du robot étant la partie la plus importante du programme, nous nous sommes aventurés trop rapidement sur la partie programmation (partie modélisation trop vite délaissé) ce qui constitue plausiblement le point faible de notre programme.

Merci de votre attention

Arbre représentatif des behaviours s'exécutant selon le contexte