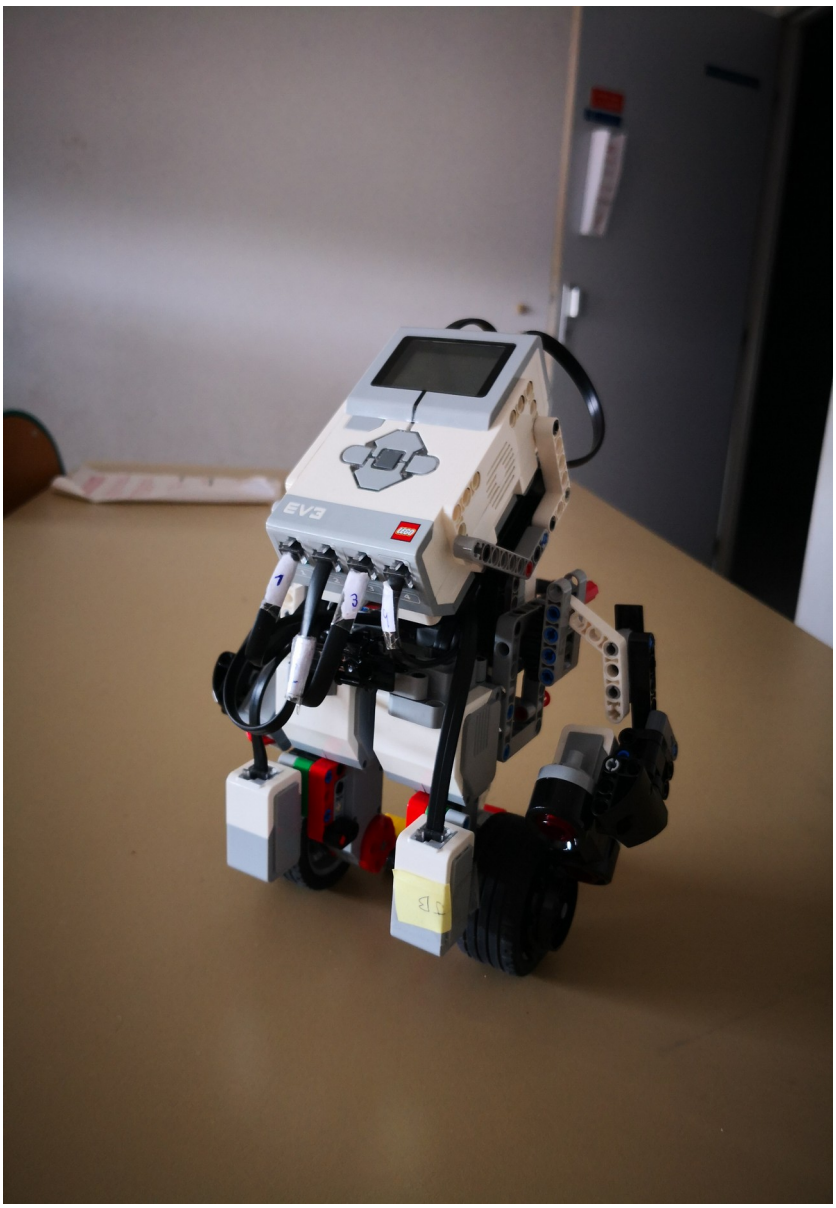
MINDSTORMS EV3

Documentation GyroBoy

Par BOURGEOIS Julien et ISAMBERT-PAYET Steven



Sommaire

I. Fonctionnement du Robot

Le GyroBoy se présente sous la forme d'un robot Lego muni d'un support lui permettant de rester droit lors de son lancement, d'un câble USB pour charger divers programme, et d'un chargeur.

1. Préparer le robot

Pour charger un programme il vous faudra posséder Eclipse (avec Java 7) et avoir le plugin leJos (http://www.lejos.org/ev3.php).
Une fois le plugin installé, branchez le robot allumé et chargez le programme en cliquant sur le bouton comme ci-dessous.

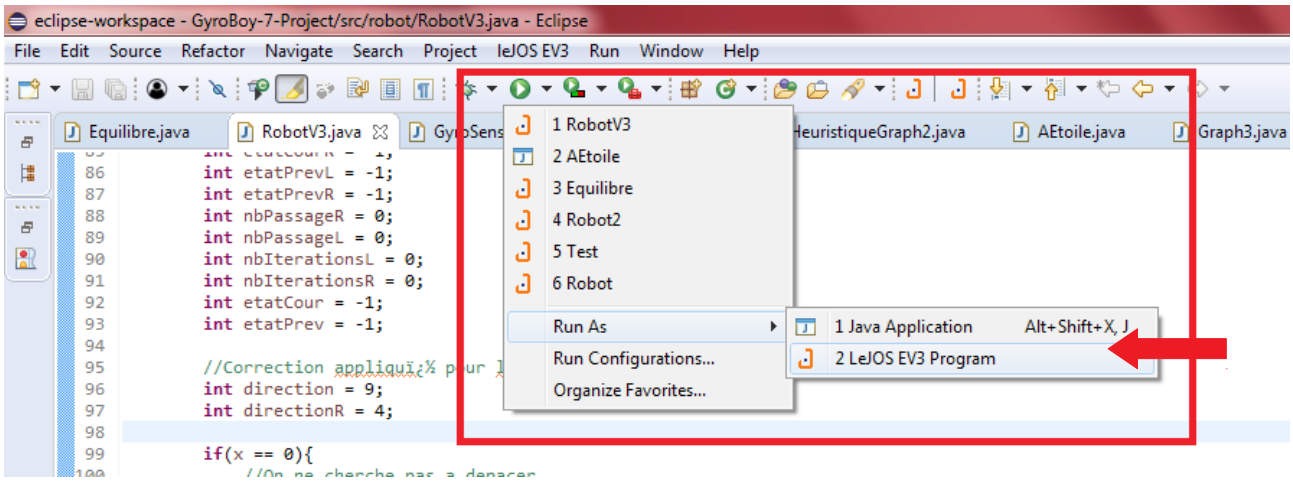


Illustration 1: Screen du bouton d'upload

Il vous faudra charger le programme RobotV3 dans le robot afin de sélectionner le mode que vous souhaitez lancer.

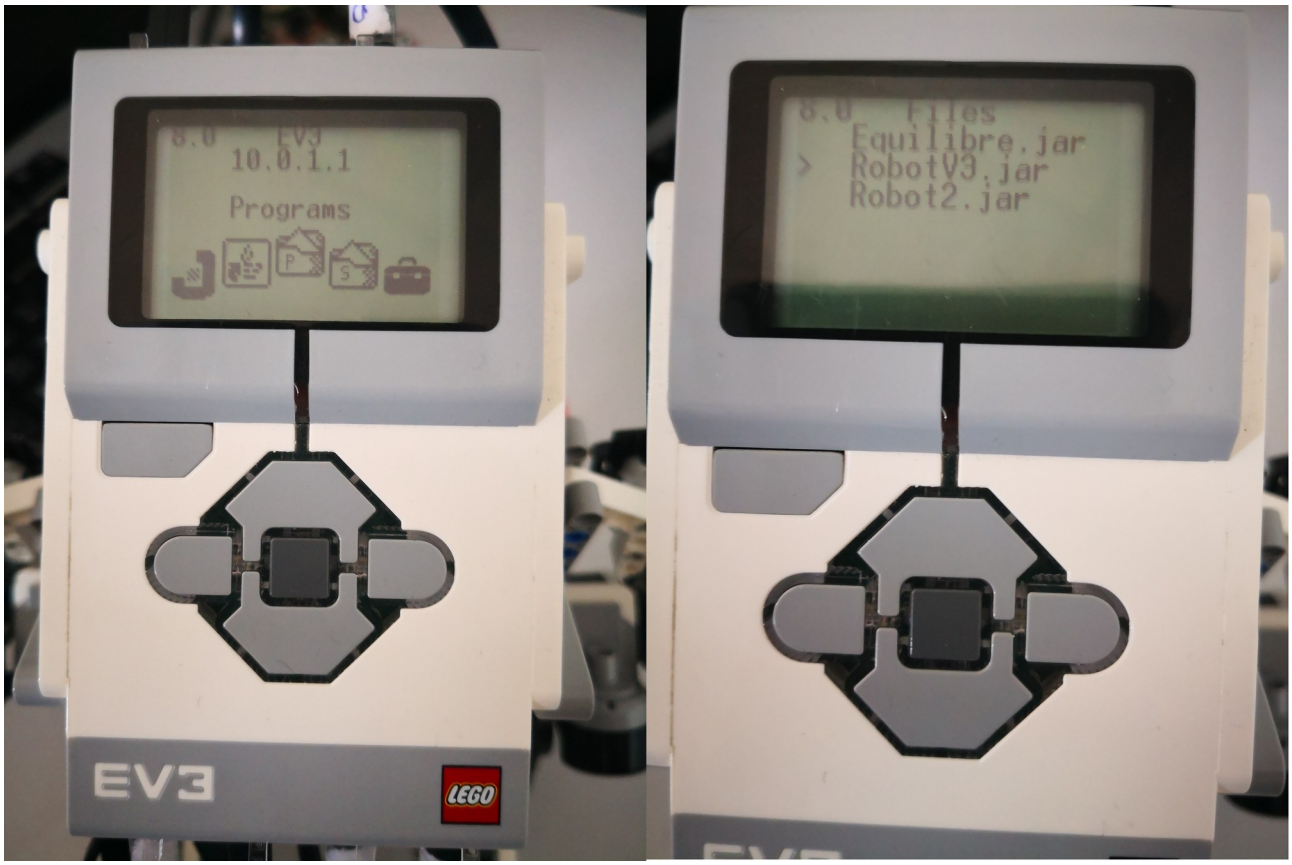


Illustration 2: Menu du Robot EV3

2. Le lancer

Une fois le programme chargé il vous faudra placer le robot avec son support sur le circuit.
Quand le robot est en place allumez le et lancez le programme RobotV3.

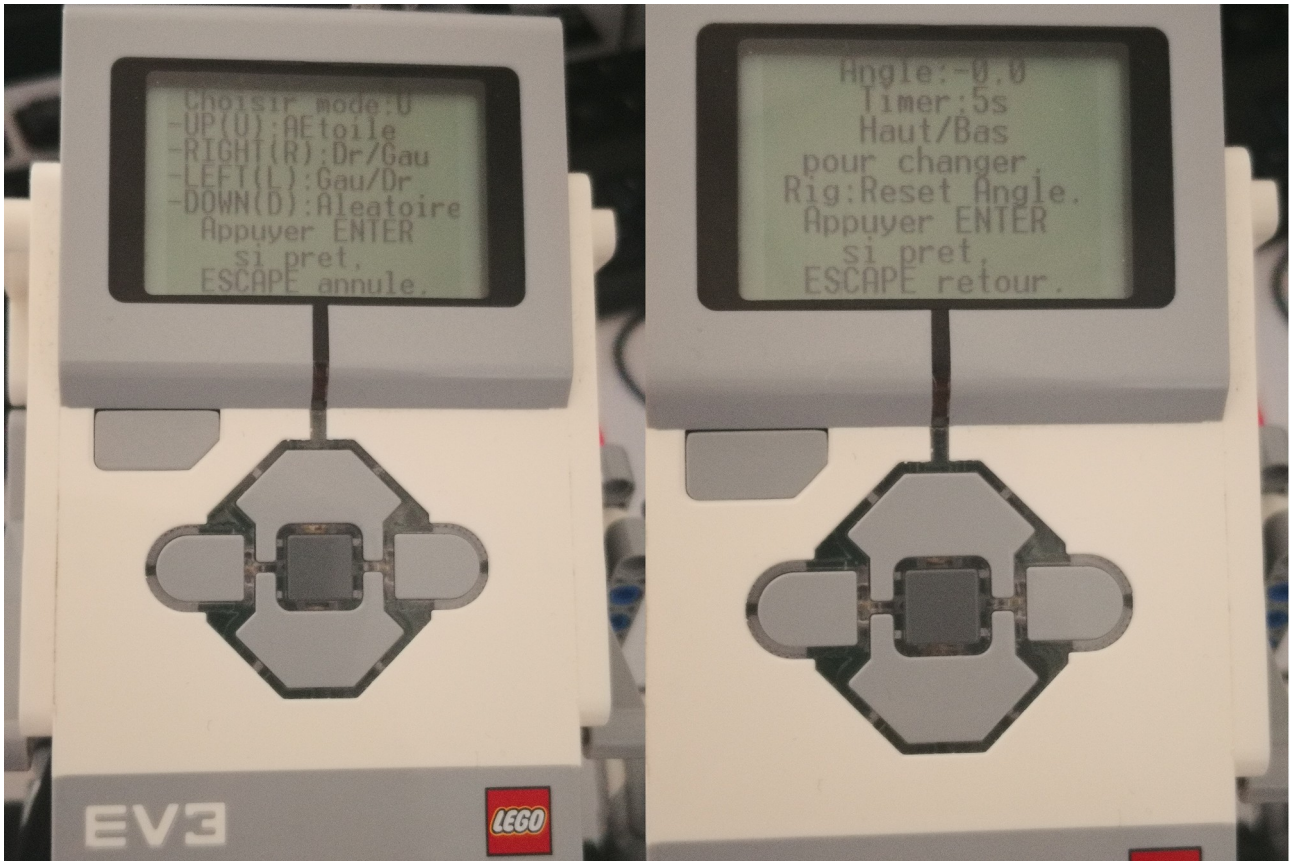


Illustration 3: Photo du menu du programme

Sélectionnez le mode que vous voulez lancer en appuyant sur le bouton correspondant.
Un menu de paramétrage va apparaître, vous pourrez changer les réglage ou bien confirmé en appuyant sur le bouton Entrer (bouton central).

Le robot va ensuite se lancer. Il émettra un beep sonore a chaque entrée/sortie de carrefour.

Important : Il faut noté que pour que le AEtoile fonctionne, le circuit devra être comme sur la photo ci-dessous, ou bien référez vous a la partie 2.3 pour modifier le code en fonction de votre circuit.

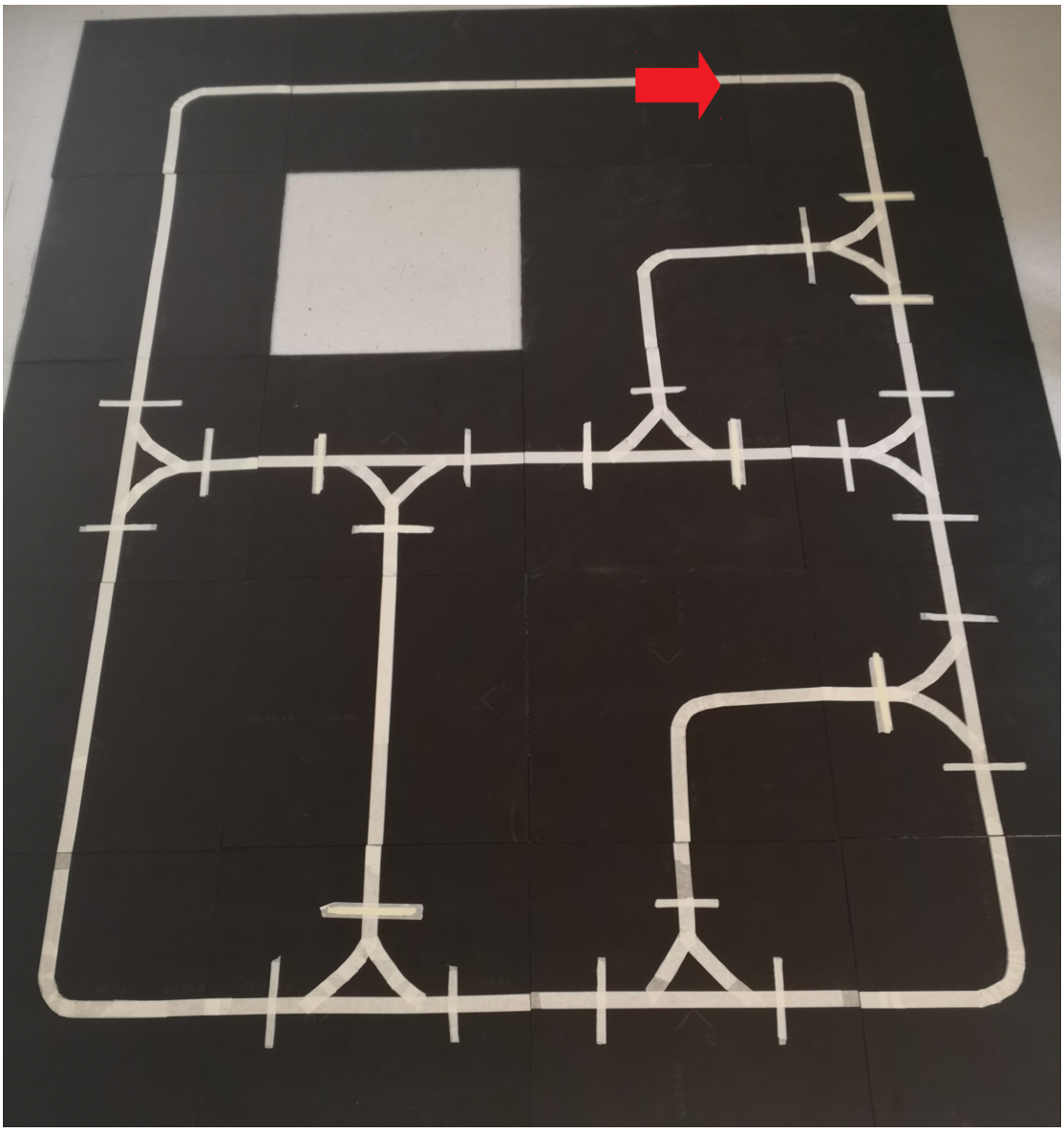


Illustration 4: Photo du circuit utilisé avec emplacement du robot

3. Stopper le robot

Pour stopper le robot, appuyez une fois sur Entrée pour stopper le mode carrefour (si jamais il a bugué) puis sur le bouton Escape (bouton en haut a gauche du robot) pour stopper totalement le programme.

Indication d'utilisation :

Veuillez noter que le robot subit des pertes d'équilibre de temps en temps dû a un défaut du gyroscope. Ceci peut entraîner des sortie de circuit et des problèmes au niveau des carrefours.

4. Montage du robot

Les instructions de montage peuvent être retrouvé a l'adresse suivante : http://robotsquare.com/wp-content/uploads/2013/10/45544_gyroboy.pdf

II. Présentation du code

1. Code de l'équilibre

L'équilibre a été développé en suivant le principe de l'algorithme PID qui est un algorithme correcteur, il va prendre un point de référence (ici l'angle de départ) et calculer la vitesse que les roue vont prendre pour corriger l'angle et revenir droit.

La partie sur l'équilibre est dans un fichier nommée Equilibre, situé dans le package robot.
On y retrouve plusieurs fonctions, les première sont les fonctions setVitesse(double x) et setDirection(int x) qui vont permettre respectivement de donnée une vitesse et une rotation au robot.

On retrouve ensuite la fonction run() qui permet de lancer le Thread de l'équilibre. Voici quelques parties du code importantes :

ligne 98 : *Thread.currentThread().setPriority(**MAX_PRIORITY**);*
Permet de faire en sorte que l'équilibre reste prioritaire sur toutes autres fonctions qui serait exécuté.

Ligne 126 : *puissance = (0.08 * vitesseMoteur) + (0.12 * angleMoteur) + (0.8 * vitesseAngulaire) + (15 * angle);*
Cette partie du code permet d'ajuster les coefficient correcteur d'équilibre, a notée que ceux-ci sont les coefficients optimaux pour le robot tel qu'il est d'origine, si vous le modifier sont équilibre sera également modifié, il faudra donc réajuster les coefficients en question.

Ligne 137 :
```
if(pret){
        if(differenceCourante<0 && compteur%2==0)
            directionC=direction-3;
        else
            directionC=direction;
        //Micro correction de la trajectoire afin de permettre au robot de se deplacer droit
        droite.setPower((int)(puissance)-directionC);
        gauche.setPower((int)(puissance)+directionC);
    }
```

Cette partie du code permet dans un premier temps de corriger un défaut du robot qui avait un décalage entre le moteur droite et le moteur gauche ce qui faisait qu'il avait tendance a tourner a droite et dans un second temps cela nous permet d'appliquer une rotation au robot.

2.Le suivie de ligne

Le code du suivie de ligne se situe dans le fichier RobotV3 situé dans le package robot. L'objectif est que le robot se déplace sur un circuit de lignes tracé au sol composé de virage, de ligne droite et de carrefour.

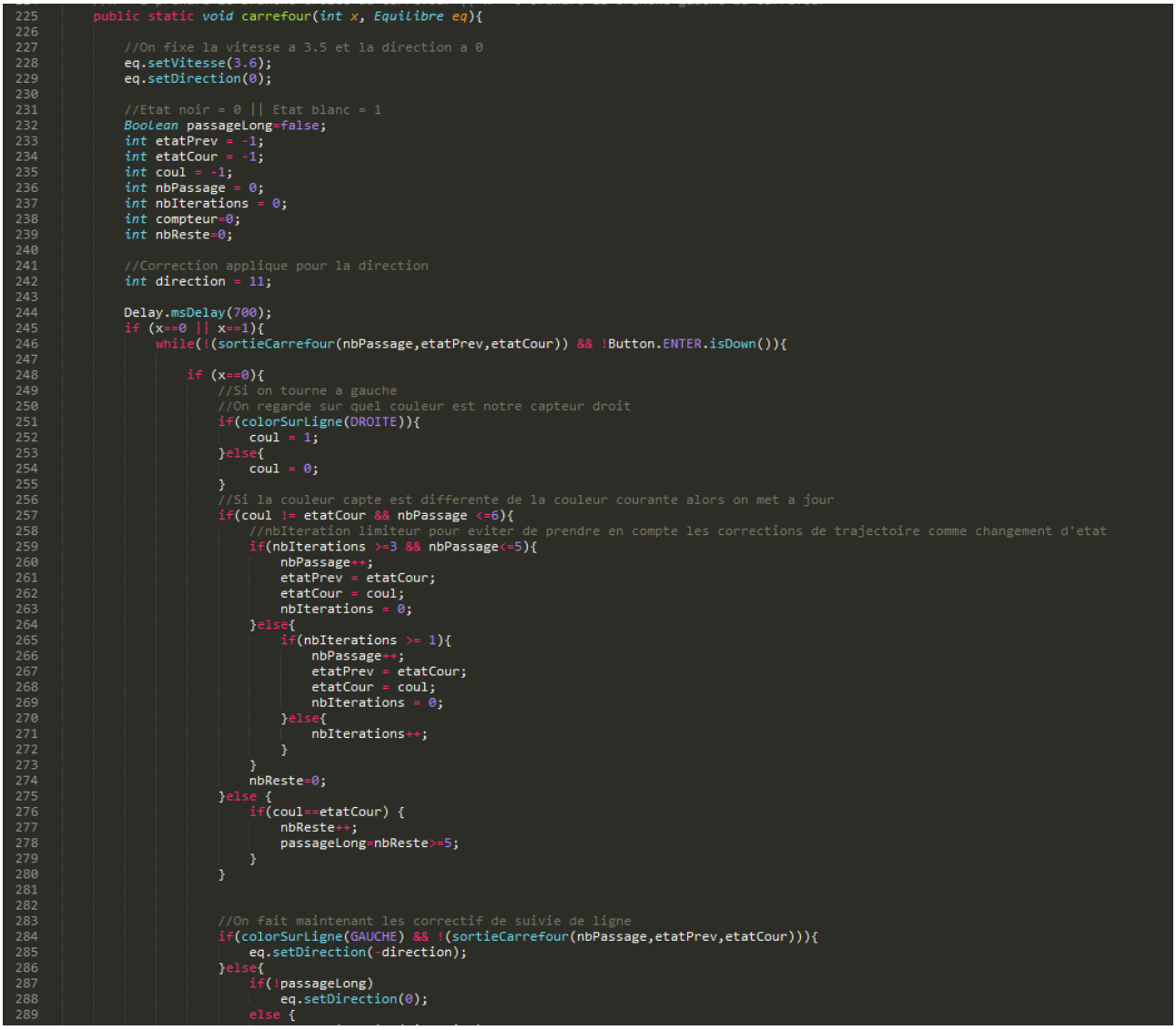
La première fonction importante rencontré est la fonction sensLigne(). Elle va lire sur les capteur de couleurs droit et gauche s'ils détectent une ligne. Si une ligne est détecté a gauche ou a droite il renverra respectivement 1 ou 2, si les deux capteur captent la ligne il renverra 3 sinon 0. Cela nous permettra dévaluer la situation dans laquelle se trouve le robot par rapport au circuit.

On retrouve ensuite une fonction demiTour() permettant au robot d'effectuer un demiTour tout en restant sur la ligne.

On a ensuite la fonction carrefour(int x) et sortieCarrefour().

Sortie carrefour nous indiquera si on peux sortir ou nous de la fonction carrefour.

La fonction carrefour est très importante, elle prend en paramètre la direction vers laquelle on veux se diriger.
(0 ou 1 → GAUCHE ou DROITE)



```
225     public static void carrefour(int x, Equilibre eq){
226
227         //On fixe la vitesse a 3.5 et la direction a 0
228         eq.setVitesse(3.6);
229         eq.setDirection(0);
230
231         //Etat noir = 0 || Etat blanc = 1
232         Boolean passageLong=false;
233         int etatPrev = -1;
234         int etatCour = -1;
235         int coul = -1;
236         int nbPassage = 0;
237         int nbIterations = 0;
238         int compteur=0;
239         int nbReste=0;
240
241         //Correction applique pour la direction
242         int direction = 11;
243
244         Delay.msDelay(700);
245         if (x==0 || x==1){
246             while(!(sortieCarrefour(nbPassage,etatPrev,etatCour)) && !Button.ENTER.isDown()){
247
248                 if (x==0){
249                     //Si on tourne a gauche
250                     //On regarde sur quel couleur est notre capteur droit
251                     if(colorSurLigne(DROITE)){
252                         coul = 1;
253                     }else{
254                         coul = 0;
255                     }
256                     //Si la couleur capte est differente de la couleur courante alors on met a jour
257                     if(coul != etatCour && nbPassage <=6){
258                         //nbIteration limiteur pour eviter de prendre en compte les corrections de trajectoire comme changement d'etat
259                         if(nbIterations >=3 && nbPassage<=5){
260                             nbPassage++;
261                             etatPrev = etatCour;
262                             etatCour = coul;
263                             nbIterations = 0;
264                         }else{
265                             if(nbIterations >= 1){
266                                 nbPassage++;
267                                 etatPrev = etatCour;
268                                 etatCour = coul;
269                                 nbIterations = 0;
270                             }else{
271                                 nbIterations++;
272                             }
273                         }
274                         nbReste=0;
275                     }else {
276                         if(coul==etatCour) {
277                             nbReste++;
278                             passageLong=nbReste>=5;
279                         }
280                     }
281
282
283                     //On fait maintenant les correctif de suivie de ligne
284                     if(colorSurLigne(GAUCHE) && !(sortieCarrefour(nbPassage,etatPrev,etatCour))){
285                         eq.setDirection(-direction);
286                     }else{
287                         if(!passageLong)
288                             eq.setDirection(0);
289                         else {
```

Illustration 5: Screen du code Carrefour (1/2)

```
                    else {
                        eq.setDirection(direction);
                        if(nbReste>=6)nbReste=0;
                    }
                }
            }else{
                //Si on tourne a droite
                //On regarde sur quel couleur est notre capteur gauche
                if(colorSurLigne(GAUCHE)){
                    coul = 1;
                }else{
                    coul = 0;
                }
                //Si  la couleur captï¿½ est diffï¿½rente de la couleur courante alors on met a jour
                if(coul != etatCour && nbPassage <=6){
                    if(nbIterations >= 3 && nbPassage <=5){
                        nbPassage++;
                        etatPrev = etatCour;
                        etatCour = coul;
                        nbIterations = 0;
                    }else{
                        if(nbIterations >= 1){
                            nbPassage++;
                            etatPrev = etatCour;
                            etatCour = coul;
                            nbIterations = 0;
                        }else{
                            nbIterations++;
                        }
                    }
                    nbReste=0;
                }else {
                    if(coul==etatCour) {
                        nbReste++;
                        passageLong=nbReste>=5;
                    }
                }
                //On fait maintenant les correctif de suivie de ligne
                if(colorSurLigne(DROITE) && !(sortieCarrefour(nbPassage,etatPrev,etatCour))) {
                    eq.setDirection(direction);
                }else{
                    if(!passageLong)
                        eq.setDirection(0);
                    else {
                        eq.setDirection(-direction);
                        if(nbReste>=7)nbReste=0;
                    }
                }

            }
            compteur=(compteur+1)%3;
            Delay.msDelay(75);
        }

        //Une fois sur l'embranchement final on laisse avancer le robot pour qu'il sorte de la double ligne
        if(x==0){
            eq.setDirection(direction);
        }else{
            eq.setDirection(-direction);
        }
        Delay.msDelay(150);
        eq.setDirection(0);
        Delay.msDelay(550);

    }else{
        Sound.playTone(800, 10, 10);
        System.out.println("/!\\ ERREUR CARREFOUR /!\\ \n\n\n");
```

Illustration 6: Screen du code Carrefour (2/2)

La dernière fonction importante de cette partie est la fonction suiveurDeLigne().

Cette fonction a pour but de maintenir le robot sur la ligne, elle va déterminer si des corrections de trajectoire sont nécessaire et si on se situe dans un cas ou l'on doit faire un demi-tour ou prendre un carrefour en fonction de la liste de trajectoire qu'elle reçoit.

```
363     public static void suiveurDeLigne(){
364
365
366         //Ensemble de nos variables permettant de fixer la vitesse et la direction
367         double vitesse = 3.4;
368         double vitesseVirage = 3.7;
369         double vitesseLigne = 3.7;
370         int direction = 0;
371         int directionVirage = 6;
372         int nbPassageVirageDroite = 0;
373         int nbPassageVirageGauche = 0;
374         int nbPassageLigneDroite = 0;
375
376         //Variable pour le carrefour
377         int leftright = -1;
378         int par = 0;
379
380         //Booleen d'arret
381         boolean onContinu = true;
382
383         //Lancement de l'equilibre avec une vitesse initiale pour demarrer le circuit
384         eq.start();
385         eq.setVitesse(vitesse);
386
387         while(!Button.ESCAPE.isDown() && onContinu){
388
389             switch(sensLigne()){
390
391                 case 0: //Case ou aucun des capteur ne capte la ligne on avance
392                         //Remise a 0 des compteur et de la direction
393                         nbPassageVirageDroite = 0;
394                         nbPassageVirageGauche = 0;
395                         direction=0;
396                         //Vitesse fixï¿½ a 3.4 au depart
397                         if(nbPassageLigneDroite == 0) {
398                             vitesse = vitesseLigne;
399                         }
400                         eq.setVitesse(vitesse);
401                         Delay.msDelay(15);
402                         eq.setDirection(direction);
403                         break;
404                 case 1: //Cas ou seul le capteur gauche capte la ligne
405                         //Remise a 0 des compteur
406                         nbPassageLigneDroite = 0;
407                         nbPassageVirageDroite = 0;
408                         //Vitesse fixï¿½ a 2.2 pour les virages et direction a 8
409                         vitesse = vitesseVirage;
410                         if(nbPassageVirageGauche == 0){
411                             direction = directionVirage;
412                         }
413                         //Augmentation incrï¿½mentale de la direction
414                         nbPassageVirageGauche++;
415                         if(nbPassageVirageGauche<=10){
416                             direction+=2;
417                         }
418                         eq.setVitesse(vitesse);
419                         Delay.msDelay(15);
420                         eq.setDirection(-direction);
421                         break;
422
423                 case 2: //Cas ou seul le capteur droite capte la ligne
424                         //Remise a 0 des compteur
425                         nbPassageVirageGauche = 0;
426                         nbPassageLigneDroite = 0;
427                         //Vitesse fixï¿½ a 2.2 pour les virage et direction a 8
```

Illustration 7: Screen du code SuiveurDeLigne (1/2)

```
427                     //Vitesse fixï¿½ a 2.2 pour les virage et direction a 8
428                     vitesse = vitesseVirage;
429                     if(nbPassageVirageDroite == 0){
430                         direction = directionVirage;
431                     }
432                     //Augmentation incrï¿½mentale de la direction
433                     nbPassageVirageDroite++;
434                     if(nbPassageVirageDroite<=10){
435                         direction+=2;
436                     }
437                     eq.setVitesse(vitesse);
438                     Delay.msDelay(15);
439                     eq.setDirection(direction);
440                     break;
441 
442                 case 3: //Cas ou es deux capteur captent la ligne on s'arrette
443                     nbPassageLigneDroite = 0;
444                     nbPassageVirageDroite = 0;
445                     nbPassageVirageGauche = 0;
446                     //Reinitialisation de la direction
447                     eq.setDirection(0);
448                     //Recuperation de la direction
449                     leftright = listDirection.get(par).intValue();
450                     par++;
451                     if(leftright!=-1){
452                         if(leftright == 2){
453                             demiTour(eq);
454                         }else{
455                             //Marqueur sonor pour indiquï¿½ l'entrï¿½e dans un carrefour
456                             Sound.buzz();
457                             carrefour(leftright,eq);
458                             Sound.buzz();
459                         }
460                     }else{
461                         eq.setVitesse(0);
462                         eq.setDirection(0);
463                         onContinu = false;
464                         Sound.beepSequenceUp();
465                         Delay.msDelay(100);
466                         Sound.beepSequenceUp();
467                         Delay.msDelay(100);
468                     }
469                     break;
470 
471                 default : //Ne sortira jamais de [0;3] mais si c'est le cas alors stop vitesse/direction et beep sonore
472                           eq.setVitesse(0);
473                           Delay.msDelay(15);
474                           eq.setDirection(0);
475                           Sound.beepSequenceUp();
476                           Delay.msDelay(50);
477                           Sound.beepSequenceUp();
478                           Delay.msDelay(50);
479                           Sound.beepSequenceUp();
480                           Delay.msDelay(50);
481             }
482 
483             //Frequence d'ï¿½chantillonage fixï¿½ pour les capteurs
484             Delay.msDelay(75);
485         }
486 
487     }
488 
```

Illustration 8: Screen du code SuiveurDeLigne (2/2)

On a à la suite de cette fonction l'ensemble des instanciation. Ces fonctions ont pour objectif de déterminer les directions a prendre pour se balader sur le circuit (soit de façon aléatoire soit en résolvant un problème avec le AEtoile).

3. Le code de l'IA

Le code de l'IA est divisé en deux parties : la partie Graph et la partie AEtoile.

3.1. Partie Graph

Le graph est construit comme ceci :

-Les sommets sont les lignes entre les intersections,

-la direction du robot est 0 si il va vers (selon la carte) le Nord ou l'Ouest sinon 1,

-si il y a ambiguïté dans la direction (comme dans le cas du sommet 2 dans l'image ci-dessous) on choisi selon le cas Nord/Sud,

-le coût est décidé selon le cas du circuit et du robot (ici vu que c'est par case, on fait le coût selon ce qui il y a : 1 pour une ligne droite, 2 pour un virage, 3 pour une intersection. Sur un sommet, il y a juste une intersection.).

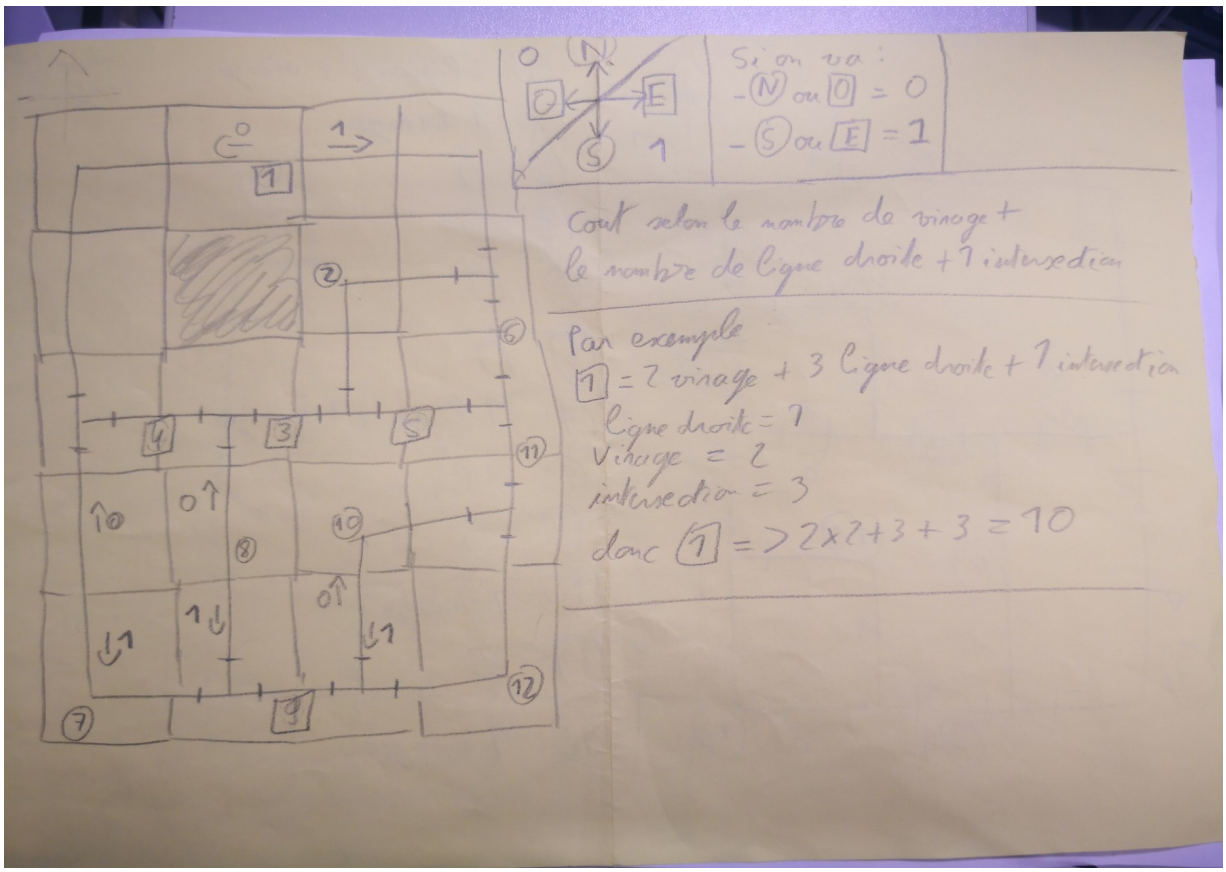


Illustration 9: Schéma de construction d'un graph

3.1.1. La classe Sommet

La classe permet de pouvoir représenter l'état d'un sommet de Graph.
Il contient 7 paramètres :

-le nom du sommet (en int pour simplifier),

-le coût de base g du sommet (=0),

-le coût total gh du sommet (utile pour le AEtoile),

-la liste de fils du sommet pour pouvoir créer les fils,

-la liste de direction pour sauvegarder le tracé que le robot doit faire,

-la direction qui est la direction fait à l'intersection précédente pour arriver sur le sommet,

-le sens du robot sur le sommet.

Il est initialiser par son constructeur Sommet avec 3 paramètres :

-le nom,

-la direction,

-le sens.

3.1.2. L'interface Graph

L'interface permet de pouvoir représenter le circuit sous forme d'une liste de voisin et de savoir dans quel direction il va par la suite.
Il contient contient deux fonctions à implémenter :

-whereDoYouCome, qui prend deux paramètres : le numéro du grand-père et le numéro du père, permet de savoir ,quand on passe une intersection , dans quel sens est le robot pour la suite,

-voisin, qui prend deux paramètres : le numéro du sommet actuelle et la direction du robot dans le sommet actuelle, permet de savoir quels sont les voisins du sommet dans la direction actuelle (donc à l'intersection suivante).

3.1.3. L'interface Heuristique

L'interface permet de fixer le coût de chaque sommet.

Il est composé d'une seule fonction à implémenter :

-fonction, qui prend un seul paramètre : le sommet, et fixe le coût selon le nom du sommet.

3.2. Partie AEtoile

3.2.1. La fonction creationFils

La fonction permet de créer la liste de fils du sommetPere à insérer dans la liste d'attente.

Il prend 4 paramètres :

-Le graph pour connaître les fils,

-Le sommetPere qui sera avec le graph,

-Le cout qu'il faut rajouter pour chaque sommet,

-L'heuristique qui permet de déterminer le coût final du sommet fils.

3.2.2. La fonction insererLesFils

La fonction permet d'insérer les fils crées qui ne sont pas dans la liste vu dans la liste d'attente.

Il prend 3 paramètres :

-La liste d'attente précédente servant de base,

-La liste des fils à insérer dans la liste,

-La liste vu pour filtrer les fils déjà vu, la fonction estPresent l'utilise.

3.2.3. La fonction aEtoile (avec sa fonction initAEtoile)

La fonction permet d'avoir une liste de direction pour aller le plus efficacement (efficacité selon l'heuristique) d'un point de départ à un point d'arrivé.

Il utilise une variable global 'dernier sens' pour savoir dans quel sens il était à la fin. (Car une fonction ne peut renvoyer deux résultats)

Il prend 5 paramètres : (le début de la fonction étant à initAEtoile)

-le graph pour connaître le circuit,

-le sommet de départ,

-le coût,

-le but pour terminer la fonction,

-l'heuristique.

3.2.4. La fonction chemin

La fonction permet de choisir un chemin, entre le point de départ et le point d'arrivé dans un sens de départ puis dans l'autre qui demande un demi-tour, selon leur coût.

Il prend 5 paramètres :

-Le graph pour connaître le circuit,

-le nom du sommet de départ,

-le nom du sommet de fin,

-l'heuristique,

-le sens de départ du robot.

3.2.5. La fonction sommetPetitFilsConstruct

La fonction permet de construire en avance les petits fils d'un sommet.

Il prend 3 paramètres :

-Le graph pour connaître le circuit,

-le sommet de départ,

-la liste initialisée de petits fils.

3.2.6. La fonction parcoursVictimes (initialisé par mainProgram)

La fonction permet de produire le chemin le plus optimisé pour sauver tous les victimes à partir d'un point de départ, un nombre de victime transportable et une liste d'hôpitaux.

Il prend 7 paramètres :

-Le nombre de victime transportable par le robot,

-le nom du sommet de départ,

-La liste des noms de sommet contenant des hôpitaux,

-La liste des noms de sommet contenant les victimes,

-le graph contenant le circuit,

-l'heuristique,

-le sens de départ.