





Fiche guide 5	TS SI		P.P.E. Robot suiveur de ligne	 académie d'Orléans-Tours Éducation nationale enseignement supérieur recherche  Lycée Polyvalent PIERRE EMILE MARTIN
Conception	4h			
 Lycée Polyvalent PIERRE EMILE MARTIN	Programme du robot élève [Partie suivi de ligne]			

Nom(s) :	Classe :	Groupe :
----------	----------	----------

Objectifs : Traduire en langage C les algorithmes de la fonction FPl « Traiter » réalisés précédemment et les intégrer à la partie existante du programme du robot à réaliser.

Matériel

Robot de test. Piste.

Logiciel


CodeVision AVR V1. Répertoire du projet (ROBSVL_x_x_x). x_x_x : dernière version du projet.

Documentation

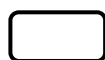
Dossier technique « Robot Elève ».

Sources des fichiers RB_Eleve.c et RB_Eleve.h

Le présent document et la documentation sont téléchargeables sur le site WebGE à l'adresse <http://p.mariano.free.fr/> (rubrique PPE)

 : Dossier technique « Robot Elève ».

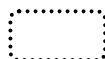
Rappel des conventions utilisées dans les schémas fonctionnels



Fonction matérielle



Energie



Fonction logicielle





Signal porteur d'une information




Variable logicielle

Appel prof

Demandez à voir la mise en œuvre du robot de test avant de lire ce document.

 Lycée Polyvalent PIERRE EMILE MARTIN	FG5	Programme Robot_E		PPE ROBOT SUIVEUR DE LIGNE	1
---	-----	-------------------	---	----------------------------	---

A) Introduction

Pour sa commande, **votre robot** sera équipé d'une carte à microcontrôleur identifiée par le sigle « **SISI** ». SISI sera informée de la position du robot par rapport à la ligne par une carte « Capteur Ligne » à cinq cellules infrarouge et commandera les deux moteurs par l'intermédiaire de deux cartes « Puissance ». L'énergie électrique nécessaire à l'ensemble sera fournie par deux accumulateurs Lipo de 7,2V montés en série. La charge de ces accumulateurs sera « surveillée » par une carte « Alimentation ». Une carte interface homme machine « IHM » vous permettra d'effectuer différents tests et réglages (robot est à l'arrêt). (Voir le )

Le **logiciel à implanter** dans le microcontrôleur de SISI se compose de deux parties :

- Une **partie « Prof »** correspondant au fichier source **RB_x_x_x.c** (x_x_x représente la dernière version du logiciel). Cette partie du programme permet d'effectuer des test et des régages sur le robot.
- Une **partie « Elève » (à compléter)** correspondant aux fichiers **RB_Eleve.c** et **RBELEVE.h**. Cette partie du programme doit assurer le déplacement du robot sur la ligne et la gestion de la balle.

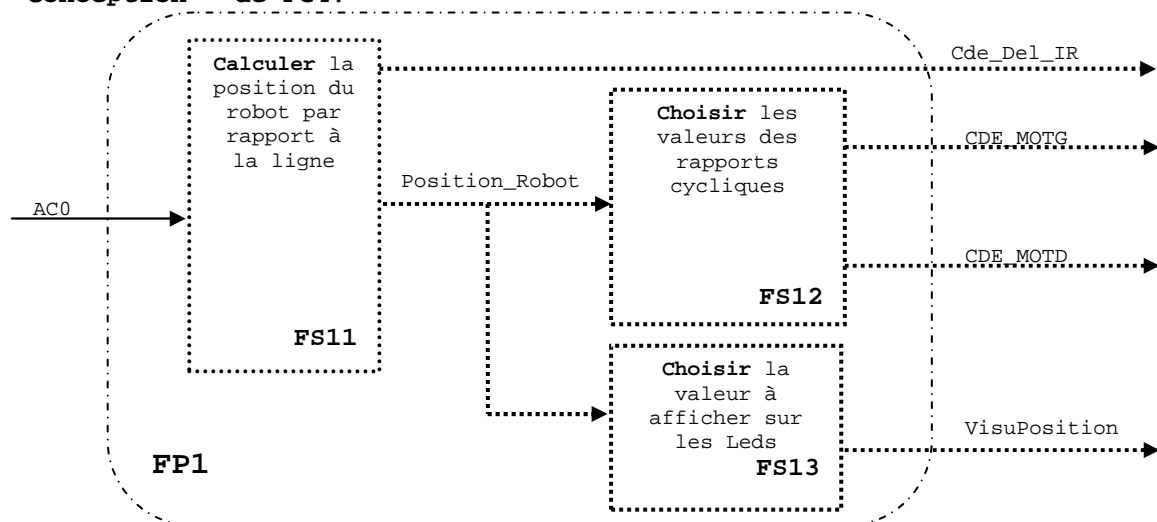
Ces deux parties sont réunies au sein d'un projet CodeVisionAVR accessible par le fichier **ROBSVL_x_x_x.prj**.

Votre travail consiste à compléter les fichiers **RB_Eleve.c** et **RBELEVE.h** en utilisant les résultats obtenus lors des séances précédentes (fiches guide 1 à 4).

Ce document concerne uniquement le **déplacement du robot**. Il doit vous guider dans l'écriture du **code C** correspondant aux algorithmes de la fonction FP1 « Traiter l'information » élaborés précédemment.

A1) Rappel : description fonctionnelle de la partie « Elève »

On rappelle ci-dessous le schéma fonctionnel de la fonction **FP1 « Traiter l'information »** étudiée dans la fiche guide 4. Cette fonction, élaborée pour le robot MrLineTiny a été transposée à votre robot dans la partie « C) Conception » de FG4.



FP1 constitue la sous-partie « Déplacement du robot » au sein de la partie « Elève ».

L'étude des fonctions :

- FS11 « Calculer la position du robot par rapport à la ligne »,
- FS12 « Choisir les valeurs des rapports cycliques », et
- FS13 « Choisir la valeur à afficher sur les Leds »

abordées dans la fiche guide 4 vous à permis d'élaborer les algorithmes correspondants pour le robot à réaliser.

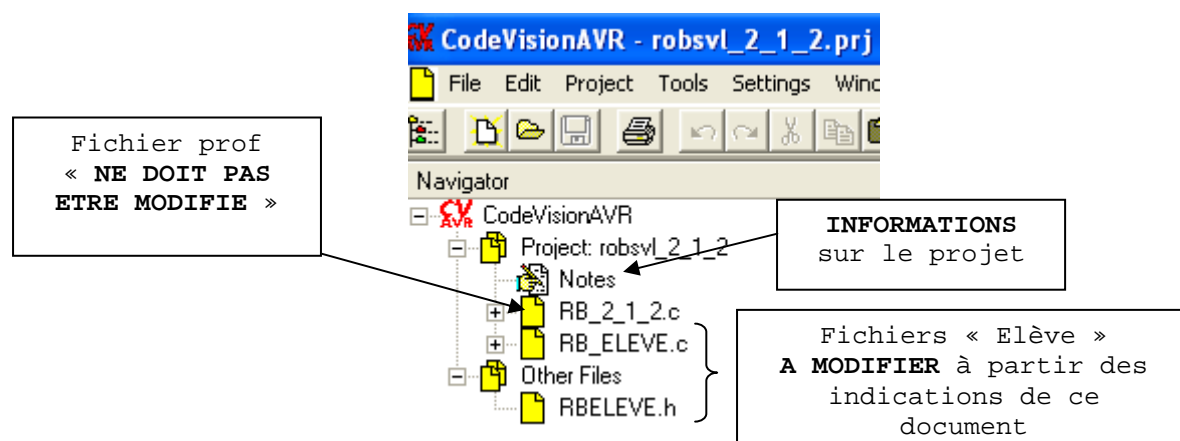
Ce sont ces algorithmes que vous allez programmer en langage C pour les intégrer au projet ROBSVL_x_x_x.

A2) Organisation structurelle du projet ROBSVL_x_x_x

Appel prof

Le projet **ROBSVL_x_x_x**, destiné à un microcontrôleur ATME8 8 bits, est développé avec le logiciel CVAVR. Les fichiers à modifier sont situés dans le répertoire ROBSVL_x_x_x (voir prof pour installer ce répertoire sur votre PC ainsi que les bibliothèques compilés auxquelles il fait référence).

Un double-clic sur ROBSVL_x_x_x.prj ouvre CVAVR (la version x_x_x peut être différente de celle apparaissant ci-dessous).



B) Programmation de la fonction FP1 « Traiter » en langage C

B1) Programmation de FS11 « Calculer la position du robot par rapport à la ligne »

Cette fonction a été étudiée dans le détail dans la fiche FG4. Sa description (rôle, entrées, sortie) est rappelée dans le fichier RB_ELEVE.c.

Q1) Complétez le programme (remplacez ????) de FS11 (situé dans le fichier RB_ELEVE.c) conformément aux résultats du paragraphe « C) conception » de la fiche guide FG4.

Remarque : le principe de détection de la position de votre robot par rapport à la ligne étant identique à celui retenu pour MrLineTiny, vous pouvez vous inspirer du programme de MrLineTiny. (Dossier technique MrLineTiny)

B2) Programmation de FS13 « Choisir la valeur à afficher sur les Leds »

Sur l'interface « IHM », les Leds D0 à D7 s'éclairent avec un niveau logique « 0 ».

Pendant le déplacement du robot, les Leds D0 à D4 doivent indiquer sa position par rapport à la ligne. Ceci est réalisé par la fonction FS13 située dans le fichier RB_ELEVE.c

Un extrait du code de FS13 dont la description (rôle, entrées, sortie) est rappelée dans le fichier RB_ELEVE.c est donné ci-dessous :

```
switch (Position_Robot)
{
    case HORS_LIGNE:    VisuPosition = Visu_Hors_ligne; break;          // Hors ligne
    ...
}

Ecrire_Port_I2C_Soft(Add_LED_I2C, VisuPosition) ; ...
```

Pour que le code de FS13 produise l'effet attendu :

- Il faut que les valeurs attribuées à Position_Robot dans l'énumération située dans le fichier RBELEVE.h correspondent à celles délivrées par la fonction FS11. Or ces valeurs dépendent du **sens de montage** de la carte « Capteur Ligne » sur le robot. (**Lisez la description** de Etat_Position_Robot dans le champ « VARIABLES » du fichier RBELEVE.h »

```
enum Etat_Position_Robot {HORS_LIGNE, DROITE_3PLUS, DROITE_2PLUS = 3, CENTRE_1,
DROITE_PLUS=7, CENTRE_3=14, DROITE, GAUCHE_3PLUS, GAUCHE_2PLUS=24,
GAUCHE_PLUS=28, GAUCHE = 30, CENTRE_5}
Position_Robot = HORS_LIGNE, Position_Robot_1 = GAUCHE_3PLUS;
```

Lecture d'une énumération

Dans l'énumération ci-dessus, HORS_LIGNE = 0, DROITE_3PLUS = 1, CENTRE_1 = 4 etc.

Q2) Identifiez la position du capteur sur le robot de test et **modifiez** éventuellement le contenu de l'énumération « Etat_Position_Robot » dans champ « VARIABLES » du fichier RBELEVE.h

Pour que le code de FS13 produise l'effet attendu :

- Il faut aussi que les valeurs telles que Visu_Hors_Ligne à placer dans la variable VisuPosition soient **définies** dans le fichier RBELEVE.h

```
//-----
// CONSTANTES
//-----
#define Visu_Hors_ligne    0xFF
... // à compléter
```

Q3) Remplacez 0xFF par la valeur à envoyer aux Leds pour chacune des positions que peut prendre le robot. **Compilez** le projet.

Appel prof

Pour **tester** votre solution sur le robot de test.

B3) Programmation de FS12 « Choisir les valeurs des rapports cycliques »

Pour éviter de devoir relier le robot au PC lors du changement des paramètres de commande des moteurs, la partie test et réglage du programme dispose d'une fonction permettant de régler ces paramètres directement sur le robot par l'intermédiaire de l'IHM de la carte SISI. Ces paramètres sont également sauvegardés dans la mémoire EEPROM du microcontrôleur. Ils sont ainsi disponibles à la mise sous tension de la carte. Lors de l'exécution du sous programme « SUIVI DE LIGNE » les paramètres de commande des moteurs situés en mémoire EEPROM sont placés dans un tableau à deux dimensions nommée **PWM**.

Ce tableau est organisé comme ci-dessous :

i	Comportement du robot	j	
		Paramètres	
		Moteur G	Moteur D
0	Centre	PWM[0][0]	PWM[0][1]
1	Droite		
2	Droite+		
3	Droite2+		
4	Droite3+		
5	HorsLigneD		
6	Gauche		
7	Gauche+		
8	Gauche2+		
9	Gauche3+		
10	HorligneG	PWM[10][0]	PWM[10][1]

Pour placer une valeur du tableau dans une variable, il faut écrire l'équivalent de `<nom_variable> <- PWM[i][j]`. Un extrait du code de FS12 dont la description (rôle, entrées, sortie) est rappelée dans le fichier RB_ELEVE.c est donné ci-dessous :

```
switch (EtatCdeMot)
{ // Début switch (EtatCdeMot)
  case DEPART: Arret_Moteurs();
    if ((Position_Robot == CENTRE_1)||
        (Position_Robot == CENTRE_3)||
        (Position_Robot == CENTRE_5))
    {
      Active_Moteurs();
      EtatCdeMot = CDE_CENTRE;
    }
    else EtatCdeMot = DEPART;
  break;

  case CDE_CENTRE : CDE_MOTG = PWM[0][0]; CDE_MOTD = PWM[0][1];
    if (Position_Robot == GAUCHE) EtatCdeMot = CDE_DROITE;
```

CDE_MOTG = PWM[0][0] se lit: On place dans la variable CDE_MOTG, la valeur contenue dans la cellule située à la position (0,0) du tableau PWM. Cette valeur correspond à celle que l'on a prévu d'envoyer à la commande du moteur gauche lorsque le capteur est au centre de la ligne.

Pour que le code de FS12 produise l'effet attendu :

- Il faut que les valeurs attribuées à Position_Robot dans l'énumération située dans le fichier RBELEVE.h correspondent à celles délivrées par la fonction FS11. Or ces valeurs dépendent de la position de la carte « Capteur Ligne » sur le robot. (Voir la description du champ « VARIABLES » dans le fichier RBELEVE.h ».

Normalement, cela a été fait à la Q2) et testé à la Q3) !

Pour que le code de FS12 produise l'effet attendu :

- Il faut également que la table effectuant la correspondance entre la valeur des paramètres PWM de commande des moteurs (entrés au clavier de l'IHM en %) et la valeur correspondante à placer dans les registres OCR1x soit correctement remplie.

Cette table se situe dans le fichier RBELEVE.h. Extrait ci-dessous ;

```
const unsigned char Table_Calcul_OCR1x[19][2]= {{0,0},{5,13},{10,25},...
```

Cette table se lit de la façon suivante : pour obtenir PWM = 5%, on place la valeur 13 dans OCR1x.

Q4) Remplacez les « ? » par les valeurs que vous avez calculées dans le paragraphe « C) Conception » de la fiche guide FG4.
Remarque : **Calculez** les valeurs manquantes.

Pour que le code de FS12 produise l'effet attendu :

- Il faut enfin compléter la description du graphe d'états, dont le début est rappelé ci-dessous, **en utilisant les valeurs de l'énumération** *Etat_Commande_Moteur* donnée dans le cadre qui suit.

```
// Graphe d'états
switch (EtatCdeMot)
{ // Début switch (EtatCdeMot)
  case DEPART: Arret_Moteurs();
    if ((Position_Robot == CENTRE_1)||
        (Position_Robot == CENTRE_3)||
        (Position_Robot == CENTRE_5))
    {
      Active_Moteurs();
      EtatCdeMot = CDE_CENTRE;
    }
    else EtatCdeMot = DEPART;
  break;

  case CDE_CENTRE : CDE_MOTG = PWM[0][0]; CDE_MOTD = PWM[0][1];
    if (Position_Robot == GAUCHE) EtatCdeMot = CDE_DROITE;
```

```
enum Etat_Commande_Moteur {DEPART,CDE_CENTRE, CDE_CENTRE2, CDE_CENTRE3, CDE_DROITE,
CDE_DROITE_PLUS, CDE_DROITE_2PLUS, CDE_DROITE_3PLUS, CDE_GAUCHE, CDE_GAUCHE_PLUS
,CDE_GAUCHE_2PLUS,CDE_GAUCHE_3PLUS,CDE_HORS_LIGNE_GAUCHE, CDE_HORS_LIGNE_DROITE }
EtatCdeMot = DEPART;
```

Q5) Complétez le texte du graphe d'état à partir de celui que vous avez dessiné dans le paragraphe « C) Conception » de la fiche guide FG4.
Compilez le projet et testez votre solution sur le robot.

Appel prof

Pour faire vérifier que votre robot suit bien la ligne !