IHM 2014-2015

Une première approche à Elisa-3

Dominique Jollien

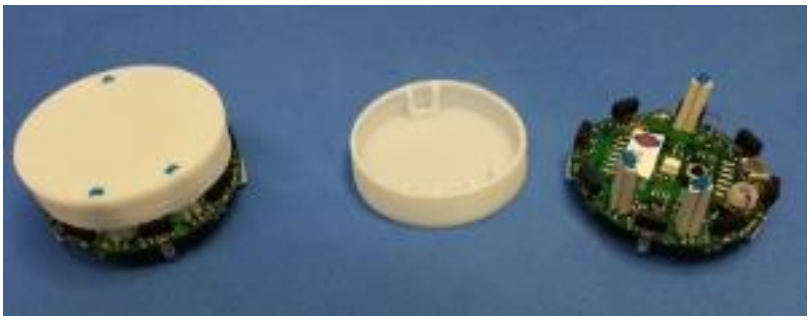
Auriana Hug

Stéphane Maillard

Marcel Sinniger

Table des matières

1 Introduction



Elisa-3 est un petit robot conçu par GCTronic et équipé notamment de capteurs, d'émetteurs infrarouges et de Leds RGB qui permettent de colorer le couvercle du robot. Elisa-3 fait partie des robots utilisés pour simuler des comportements de colonie. En effet, sa petite taille permet de mettre en place aisément un environnement pour plusieurs robots de son type.

Le but de ce projet est de pouvoir mettre en place un scénario intéressant utilisant des robots. En effet, la Maison d'Ailleurs, musée de la science-fiction situé à Yverdon-les-Bains, mettra à notre disposition un espace afin de montrer notre travail, dans le cadre de sa future exposition sur la thématique des robots. Pour notre projet, nous avons prévu un concept basé sur les caractéristiques du robot Elisa-3, que nous décrirons dans ce rapport.

En outre, ce document explique les différents outils de programmation nécessaires pour travailler avec le robot Elisa-3. De plus, les deux fonctionnalités imaginées dans notre scénario seront expliquées au niveau de leur mise en œuvre et de leur implémentation. De plus, nous partagerons dans ce document notre expérience vécue lors de ce travail.

2 Documentation officielle

Le site officiel du fournisseur décrit les différentes manières de programmer et de gérer le robot Elisa-3. Voici le lien : http://www.gctronic.com/doc/index.php/Elisa-3.

3 Description du robot

3.1 Déplacement

Pour se déplacer et tourner, le robot Elisa-3 possède deux roues indépendantes, chacune étant couplée à son propre moteur DC. Sous le robot se trouvent quatre capteurs optiques pour la détection du sol. Ils permettent d'éviter les chutes lors de l'approche d'un bord de table, par exemple. De plus, un accéléromètre sur trois axes permet au robot de connaître son orientation dans l'espace. Le déplacement du robot et la charge sont indépendants de la gravité. Le robot travaille également à la verticale et à l'envers. Sa position peut être exploitée par le programme du robot afin de changer le comportement du robot, par exemple.

3.2 Détection d'objets

Elisa-3 est équipé de 8 émetteurs infrarouges couplés avec des senseurs répartis sur le côté de l'appareil. Ils permettent notamment la détection d'objets à proximité. Dans une moindre mesure, ils peuvent également communiquer des informations avec un robot semblable.

3.3 Communication avec un ordinateur

Le robot possède aussi une antenne RF afin de communiquer avec un ordinateur. Il envoie les informations de ses capteurs et peut recevoir des instructions. En outres, le robot est équipé d'un processeur ATMEL ATmega2560 compatible Arduino. Un port micro-USB permet la connexion à un ordinateur afin de changer le code du robot. Un sélecteur à 8 positions est présent sur le PCB.

3.4 Couleurs

Une led RGB centrale permet au robot d'afficher un large choix de couleurs sur le dessus du l'appareil.

3.5 Batterie

Une batterie équipe le robot. Elle peut être chargée soit en branchant un câble dessus, soit en positionnant les deux contacts dorés sur une station adaptée.

4 Matériel utilisé

Pour ce projet, nous avons utilisé :

- 8 robots Elisa-3
- 1 antenne RF (avec son câble)
- 2 bases de recharges (avec leur câble)
- 1 environnement (l'arène)

5 Environnement de développement

5.1 Développement pour le robot

GCTronic propose plusieurs IDE pour programmer les robots. Nous allons les passer en revenue dans ce chapitre.

5.1.1 Aperçu des IDEs

5.1.1.1 AVR Studio

AVR Studio n'a jamais fonctionné sur nos machines respectives. Des problèmes de drivers sont survenus, et l'installation a écrasé la variable d'environnement « PATH » de Windows. Par conséquent, nous ne recommandons pas d'utiliser AVR Studio.

5.1.1.2 Arduino

Arduino fonctionne plutôt bien. Finalement, nous avons utilisé ce logiciel pour la programmation des robots. Malheureusement, la majorité des exemples de code pour les robots sur le site du fournisseur ont été conçus pour AVR Studio. En outres, la traduction du code est pénible, car la librairie n'est pas complétement la même pour les deux IDE.
Ainsi, la librairie "Avancée" doit être utilisée avec AVR Studio. Nous n'avons pas réussi à la faire fonctionner en utilisant Arduino.

5.1.1.3 Aseba

Ce logiciel n'a pas été utilisé lors de ce travail.

5.1.1.4 Matlab

Ce logiciel n'a pas été utilisé lors de ce travail.

5.1.1.5 Webots simulator

C'est un logiciel payant (très cher). La version démo n'a pas de réél intérêt. En effet, même pour des petits tests, on atteint rapidement les limites de la démonstration. De plus, le code à compiler n'est pas le même que pour Arduino. Par conséquent, il n'est pas possible de reprendre du code d'Arduino et de le compiler dans Webots. Pour toutes ces raisons, nous n'avons pas utilisé ce logiciel durant le projet.

5.2 Installation du logiciel Arduino

Cette page web du fournisseur explique comment installer Arduino :
http://www.gctronic.com/doc/index.php/Elisa-3#Arduino_IDE_project

Après l'installation et la configuration de ce logiciel, il est nécessaire de faire un premier test. Pour ce faire, il est possible de suivre les instructions situées juste après la section qui décrit les modifications du fichier « boards.txt ».

L'utilisation des programmes développés pendant le cours nécessite l'utilisation de la librairie côté robot basique que nous avons modifié. Celle-ci a notamment dans le fichier constants.h des valeurs différentes pour LINE_IN_THR LINE_OUT_THR.

Comme indique « *Tools=>Serial Port* », il faut choisir le port COM pour établir la connexion avec le robot, à l'aide du câble USB. Nous avons rencontré des problèmes avec les ports USB 3.0. En effet, la connexion au robot ne fonctionne pas avec ces derniers. Heureusement, nous avons constaté que ceci fonctionne avec les ports USB 2.0. Sur les ordinateurs portables de LeNovo, on peut utiliser les ports USB, identifiés en jaune (ports USB 2.0).

6 Développement du côté ordinateur avec l'antenne

Pour qu'on puisse communiquer depuis un ordinateur avec les robots à l'aide de l'antenne, il faut tout d'abord installer quelques outils. Les procédures d'installation et de configuration sont décrites sur la page du fournisseur d'Elisa-3 :
http://www.gctronic.com/doc/index.php/Elisa-3#PC_side

La mise en œuvre d'un projet « PC-Side » est pénible. C'est pourquoi nous avons préparé une version de CodeBlocks portable qui est déjà préconfigurée (voir le chapitre 9 « Contamination »). Il faut installer MinGW[1] dans le dossier C:\MinGW. Lors de l'installation, il est nécessaire de sélectionner les compilateurs C et C++ de MinGW, afin de les installer. Lorsque la procédure est terminée, il faut brancher l'antenne sur un port USB 2.0. En effet, le problème mentionné dans le chapitre précédent peut survenir.

6.1 Debugging avec l'antenne

La communication à l'aide de l'antenne permet aussi de récupérer des valeurs des capteurs des robots. En effet, les robots peuvent envoyer des valeurs au PC. Ceci permet de mieux comprendre ce qui se passe sur le robot.
Le projet précompilé du fournisseur dans le chapitre « PC side » montre comment des valeurs peuvent être échangées. Une version exécutable de ce projet se trouve dans le dossier « Debugging\Debugging_avec_l_antenne »
De plus, le sous-chapitre « Code du côté PC » du chapitre « Contamination » exploite cette fonctionnalité pour la récupération des états des batteries des robots.

[1] Lien URL : http://mingw.org

7 Rechargement

7.1 Concept

Ce robot a la faculté de se recharger de manière plus ou moins autonome. En effet, en utilisant le code fourni par le fournisseur, Elisa-3 peut normalement détecter le niveau de sa batterie, et selon la limite inférieure prévue, il peut décider d'aller à la station de recharge. Pour ce faire, du ruban noir est placé dans les positions de charge pour aider les robots à bien se placer grâce à leurs capteurs au sol. Cette fonctionnalité de recharge est importante dans notre projet, car le déroulement du scénario n'est jamais censé s'arrêter.

7.2 Arène

En effectuant les tests avec les robots pour le rechargement, nous avons constaté que le matériel, en particulier l'intégration de la plaquette avec la station de rechargement dans l'arène, joue un rôle crucial. Les robots doivent pouvoir atteindre la station de rechargement sans devoir monter sur la plaquette verte de celle-ci. En effet, le rechargement ne fonctionne pas si on l'essaie de faire monter les robots sur la plaquette verte. Celle-ci ne doit donc pas créer une différence de niveau du sol. Autrement dit, il faut intégrer la plaquette verte à l'arène.

7.3 Traits noirs sur le sol

Pour bien diriger le robot à la station de rechargement, il faut que l'environnement ait un fond blanc, et y dessiner des traits noirs (ou y coller du ruban adhésif noir) en direction de la station. Le contraste entre la couleur des traits et celle de l'arène doit être maximal.

En ce qui concerne la largeur de trait, il faut se baser sur la distance entre deux capteurs infrarouge au-dessous du robot, ce qui fait environ 1 cm. Nous avons testé des traits plus fins, mais ceux-ci ne marchent pas.

Un problème peut survenir si un robot croise un trait à l'angle droit : les robots n'arrivent pas à se positionner ; ils quittent simplement la ligne. Un robot doit croiser les traits à un angle assez faible pour qu'ils arrivent à se positionner correctement et suivre la ligne. De plus, on a le même problème si l'on crée un parcours avec les traits avec des changements d'orientation trop brusque.

Nous avons fait l'expérience suivante : nous avons imprimé des lignes noires sur une feuille blanche. Cet essai s'est montré concluant.

7.4 Algorithme

Voici en substance l'algorithme de rechargement. Pour plus de détail, il faut se référer au code.

1. Le robot se déplace aléatoirement un certain temps (ceci a été supprimé dans le code où la contamination et le rechargement ont été mis ensemble).
2. Le robot se déplace aléatoirement jusqu'à arriver sur une ligne.
3. Le robot suit la ligne jusqu'à arriver à la borne de rechargement. Il peut la suivre dans le mauvais sens. S'il passe plus de 20 secondes à la suivre, on considère qu'il est bloqué et l'algorithme recommence.
4. Le robot se charge pour avoir un certain niveau de batterie puis recule.
5. Le robot se retourne.

7.5 Code

Le programme de rechargement est une adaptation du programme de démo fourni dans la démo de base (dans sa version Arduino), disponible sur le site du fournisseur. Le code se trouve dans le dossier « Elisa3-rechargement ».

Les constantes LINE_IN_THR et LINE_OUT_THR sont utilisées comme valeurs de références pour les capteurs au sol du robot. Elles permettent de détecter les lignes noires. Elles ont été modifiées dans la version de la librairie Elisa-3 (librairie de base) que nous utilisons. Cette version modifié se trouve dans le dossier « Elisa3-librairie à utiliser ».

8 Contamination

8.1 Concept

Un jeu appelé « Contamination » a été développé. Il visualise la transmission d'une maladie dans une colonie de robots. Il y a deux états dans lesquels un robot peut se trouver :

- Victime (robot sain)
  - Le robot est sain.
  - Le robot ne se déplace pas.
- Zombie (robot contaminé)
  - Le robot est infecté.
  - Le robot se déplace de manière aléatoire.

Dès le moment où un robot « Zombie » s'approche d'un robot « Victime, la maladie est transmise au robot « Victime ». Par conséquent, ce robot devient lui aussi contaminé et change son état en celui de « Zombie ».

8.2 Code du côté PC

Le code du côté PC permet de démarrer le jeu « Contamination ». Dans ce code, il faut préciser le nombre de robots dans l'arène et tous les numéros des robots (le numéro figurant au-dessous des robots).

Le code se trouve dans le dossier « Elisa3-Contamination\PC_Side\Elisa-3_PC-SIDE_auto_select_robot\pc-side-elisa3-auto_select_robot ». C'est un projet CodeBlocks. Une version directement utilisable (sans installation) du programme CodeBlocks se trouve dans le dossier « Elisa3-Contamination\PC_Side\codeblocks ».

8.2.1 Fonctionnement du programme

Le programme effectue les tâches suivantes :

- Démarrer la communication radio avec tous les robots
  - Attendre 3 secondes (chaque 2 secondes, les robots envoient l'état de leur batterie).
  - Lire les états des batteries.
  - Terminer la communication radio.
- Exclure tous les robots qui ont un état de batterie trop bas.
- Démarrer la communication radio uniquement avec les robots qui font partie du jeu.
  - Mettre tous les robots dans l'état « Victime ».
    - Couleur bleu (bleu = 100%).

- Lister tous les robots qui font partie du jeu.
- Lasser l'utilisateur choisir le premier robot contaminé.
  - L'utilisateur saisit le numéro du robot (affiché dans la console).
- Envoyer un signal au robot choisi pour le mettre en mode « Zombie » (contaminé).
- Le robot contaminé commence à se déplacer et à infecter les autres robots.
  - Si un robot contaminé (rouge) s'approche d'un robot sain (bleu), le robot sain devient contaminé et devient contaminé.

8.2.2 Détails d'implémentation du programme

Les détails d'implémentation sont décrits dans les commentaires du code. Ce chapitre est donc consacré au concept de la transmission du « signal » depuis l'ordinateur aux robots.

GCTronic fournit déjà des fonctions pour la communication ordinateur-robot dans la librairie. A noter qu'il y a une librairie pour le côté ordinateur et une autre pour le robot. Il est par exemple très facile de mettre les couleurs aux robots depuis l'ordinateur.

Dans le cas du jeu de contamination, nous avons eu besoin d'un signal qui permet de déclencher un mode sur les robots. Pour des raisons de simplicité, et aussi parce que nous voulions modifier la librairie le moins possible, nous avons utilisé la couleur comme signal. C'est-à-dire que mettre la couleur bleu (100%) à un robot change le mode d'Elisa-3 en « Victime ». Mettre la couleur rouge (100%) démarre le mode « Zombie ». Les robots vérifient leur couleur à chaque itération de la procédure principale. Selon la couleur, ils activent un autre comportement.

Si l'on veut ajouter un autre jeu, on peut donc imaginer envoyer aussi la couleur rouge ou bleu, mais que 99% par exemple (vu que 100% est utilisé par « Contamination »), pour démarrer ce jeu. Autrement dit, on peut jouer avec des couleurs pour démarrer des différents comportements. Pour le visiteur, il n'y a pas de différence entre un rouge 100% et un rouge 99%.

8.3 Code du côté robot

L'implémentation du concept du jeu « contamination » s'est réalisé sur le côté ordinateur ainsi que sur le côté robot.

Le code se trouve dans le dossier « Elisa3-Contamination\Robot_Arduino_Project\contamination ». C'est un fichier à ouvrir avec le logiciel Arduino. Le chapitre « Installation du logiciel Arduino » explique comment procéder à l'installation et au téléchargement du code vers le robot.

8.3.1 Etat de contamination

```
unsigned int contaminated = 0;
```

En interne, les états de contamination sont représentés par un entier qui peut prendre les valeurs suivantes:

0 = neutre (le jeu de contamination n'est pas en cours)

1 = victime, le robot est sain (non-contaminé)

2 = zombie, le robot est contaminé

8.3.2 La boucle principale

Un programme Arduino pour Elisa 3 est composé d'une boucle principale au lieu d'une fonction « main » comme dans un programme C normal.

Ci-dessous, un extrait du code du « main loop » montre les fonctions importantes. Tous les fonctions (sauf mentionnée explicitement) font partie de la librairie principale fourni par le fournisseur.

```
void loop() {
...
    readBatterie();
...
    handleRFCommands();
...
    enableObstacleAvoidance();
    ...
    if (getTime100MicroSec() > 50000) {
        contamination();
    }
...
    handleMotorsWithSpeedController();
}
```

- `readBatterie`
  - Permet de calculer l'état actuel de la batterie du robot. Cette fonction doit être appelée pour que le programme du côté PC puisse toujours lire la valeur actuelle de la batterie.
- `handleRFCommands`
  - Permet de faire écouter au robot les signaux envoyés par l'ordinateur à l'aide de l'antenne radio. De plus, cette commande est nécessaire pour que le robot transmette ses états (batterie, valeurs de capteurs, …) au PC.
- `enableObstacleAvoidance`
  - Permet d'activer la détection d'obstacles.
- `getTime100MicroSec() > 50000`
  - Permet de faire attendre le robot pendant environ 5 secondes. Cette attente permet que le robot soit bien initialisé avant qu'il puisse récupérer des commandes pour le jeu de contamination.
  - Dans le fichier « adc.c » (librairie des robots) à la ligne 45 se trouve l'incrémentation de l'horloge interne. Cette variable qui représente l'horloge interne se trouve dans le fichier « variables.h / variables.c »
  - Remarque : La variable fait un « wrap around » après un certain temps. Voir les lignes 151 et 152 du fichier « variables.c »
- `contamination`
  - C'est la fonction principale qui gère le jeu de contamination. Cette fonction est décrite par la suite en détails.
- `handleMotorsWithSpeedController`
  - Cette fonction est responsable pour le bon fonctionnement de la gestion de la vitesse du moteur.

8.3.3 La fonction d'initialisation «setup »

Comme la fonction « loop », la fonction « setup » est une fonction spécifique à Ardunio. Elle est appelée avant la fonction « loop » et permet d'initialiser le robot.

8.3.4 La fonction « contamination »

```
void contamination() {

    // start game
    if (pwm_red == 0) {
        contaminated = 2;
    } else if (pwm_blue == 0) {
        contaminated = 1;
    }

    // check if a contaminated robot is nearby
    if (contaminated == 1) {
        unsigned int proximityResultLinearSum = 0;
        int i = 0;
        for(i=0; i<8; i++) {
            if(proximityResultLinear[i] < NOISE_THR) {
                proximityResultLinear[i] = 0;
            }
            proximityResultLinearSum +=
proximityResultLinear[i];
        }

        // contamination
        if (proximityResultLinearSum > 100) {
            // become a zombie
            contaminated = 2;
        }
    }

    // contamination - victime
    if(contaminated == 1) {
        setLeftSpeed(0);
        setRightSpeed(0);
        pwm_red = 255;
        pwm_green = 255;
        pwm_blue = 0;
    // contamination - zombie
    } else if (contaminated == 2) {
        pwm_red = 0;
        pwm_green = 255;
        pwm_blue = 255;
        setLeftSpeed(25);
        setRightSpeed(25);
    }
    updateRedLed(pwm_red);
    updateGreenLed(pwm_green);
    updateBlueLed(pwm_blue);

}
```

- `// start game`
  - Comme décrit dans le chapitre « Code du Côté PC », la gestion se base sur les couleurs.
  - La valeur correspondante est assignée à la variable « contaminated » selon la couleur du robot.
- `// check if a contaminated robot is nearby`
  - La première partie de ce code récupère les valeurs des huit capteurs infrarouges qui sont utilisés pour la détection d'obstacle. La comparaison des valeurs avec la constante « NOISE_THR » permet d'enlever le bruit des capteurs.
  - Ensuite, la somme de ces capteurs est comparée avec un seuil (actuellement à 100, qu'il faut adapter selon les conditions d'environnement, comme la luminosité par exemple). Si le seuil est dépassé, une rencontre entre deux robots est constatée et le robot « victime » devient un « zombie ».
- `// contamination - victime / // contamination - zombie`
  - Finalement, selon l'état du robot (« victime » ou « zombie ») la vitesse ainsi que la couleur sont attribuées.

9 Contamination et rechargement

Un programme réunissant les deux fonctionnalités (contamination et rechargement) a été réalisé. Si les robots sont en-dessous d'un niveau de batterie défini dans le code, ils exécutent le programme de rechargement, sinon celui de la contamination. Le code se trouve dans le dossier « Elisa3-contamination+rechargement/contamination _recharge-ment ». A noter que le programme utilisé sur l'ordinateur qui permet de paramétrer le jeu de la contamination n'est pas finalisé. En effet, il fait une vérification au démarrage du niveau de batterie des robots pour ne pas que l'on puisse envoyer d'ordre aux robots qui sont en phase de rechargement. Toutefois, cette vérification n'est faite qu'une seule fois, au début du programme, et évidemment les niveaux de batteries vont évoluer au cours du temps. Pour la suite, il est donc nécessaire d'élaborer une stratégie générique pour gérer les robots qui sont en train de se charger ou qui sont en train de chercher la station de rechargement.

10 Conclusion

Lors de ce travail, nous avons fait la connaissance d'Elisa-3, ce petit robot, et nous avons découvert sa librairie, qui est assez stable, mais malheureusement pas très bien documentée. A notre avis, il manque un document qui explique une marche à suivre pour pouvoir facilement commencer à développer. Nous espérons que ce rapport ainsi que la documentation au niveau du code combleront cette lacune laissée par le fournisseur.

11 Annexe

- Debugging
  - Héberge un outil qui permet de récupérer certaines valeurs de robots.
- Elisa3-Contamination
  - Code du jeu de contamination
- Elisa3-rechargement
  - Code du rechargement
- Elisa3-contamination+rechargement
  - Code réunis
- Elisa3-librairie à utiliser
  - Librairie modifié pour le côté robot
- Presentation_cahier_des_charges.pdf
  - Présentation initial du projet
- Presentation_bilan_final.pdf
  - Présentation final du projet