# Robotics Project I

David Baschung, Groupe 6

IN.2022 Robotique 2019, Cours BSc, 2nd Sem.
Université de Fribourg
david.baschung@unifr.ch

**Résumé**

Ce rapport contient une série de tests effectués sur des e-pucks 2, petits robots à 2 roues. Nous avons implémenté les controlleurs de Braitenberg, comme le Lover et l'Explorer, et diverses utilisations possibles des capteurs de proximité, des Senseurs au sol et de la caméra. Cette étude vise à ajuster les valeurs retournées par le robot et tente de les maîtriser en les ajustant dans les controlleurs. La communication inter-robots est robot-ordinateur est aussi abordée.

**Keywords:** e-puck, epuck, robot, webots, capteur, senseur, proximité, infrarouge, caméra, couleur

The use of LaTeX is mandatory for the Project I report. Apart from the examples in the appendix below, this template may not be modified. A good introduction to scientific writing is given by [1]

# Table des matières


**1 Introduction** **2**
- 1.1 Capteurs de proximité infra-rouge . . . 2
- 1.2 Capteurs au sol . . . 2
  - 1.2.1 Valeurs . . . 2
  - 1.2.2 Disposition . . . 2
- 1.3 Caméra . . . 3

**2 Comportements** **5**
- 2.1 Braitenberg . . . 5
  - 2.1.1 LOVER . . . 5
  - 2.1.2 EXPLORER . . . 5
- 2.2 Line-following . . . 5
  - 2.2.1 Comportement de base . . . 5
  - 2.2.2 Dépassement frontal . . . 6
- 2.3 Wall-following . . . 6
- 2.4 Color recognition . . . 6
  - 2.4.1 Détection . . . 6
  - 2.4.2 Implémentation en mouvement . . . 7
- 2.5 Multi-robot coordination . . . 7
  - 2.5.1 Communication : généralités . . . 7
  - 2.5.2 Alternative search . . . 7
  - 2.5.3 Simultaneous search . . . 9

**3 Conclusion** **10**

**Appendix** **12**
- Appendix A Experimental Results . . . 12
- Appendix B Source Code . . . 12
  - B.1 IR sensors calibration procedure . . . 12
- Appendix C LaTeX Examples . . . 12
  - C.1 Images . . . 12
  - C.2 Tables . . . 12
  - C.3 Listings . . . 13
  - C.4 Font Style and Text Size . . . 14
  - C.5 Enumerations and Other Lists . . . 14
  - C.6 Quotations and References . . . 14
  - C.7 FSM diagram . . . 14

# Chapitre 1

# Introduction

Dans le cadre de de notre cours de robotique, nous avons testé et implémenté plusieurs controlleurs. Nous les avons testés d'abord sur le simulateur Webots, avant de comparer les résultats sur un vrai robot. Le but du cours est d'introduire l'utilisation de Linux et la programmation en C sur le simulateur webots, avant de faire marcher de vrais robots. Diverses utilisations sont au programme, comme l'évitement d'obstacles et l'utilisation de la caméra.

## 1.1 Capteurs de proximité infra-rouge

Afin de pouvoir déterminer la situation du robot dans son entourage, nous utilisons des senseurs à rayons infra-rouges. Huit senseurs sont disposés tout autour du robot et dirigés vers l'extérieur. Chaque senseur émet un signal différent et peut déterminer dans quelle proportions il se trouve à proximité d'un obstacle, sans pour autant donner une distance exacte. Nous avons ainsi remarqué que les objets blancs réfléchissent le signal avec une plus forte intensité que les objets noirs, ou même les verts. Ces senseurs permettent donc de retourner une valeur en temps réel qui pourra être utilisée à son tour pour influencer la vitesse des roues et le comportement du robot.

## 1.2 Capteurs au sol

### 1.2.1 Valeurs

Dans le but d'éviter une chute, trois senseurs sont disposé sous le robot. Ils retournent chacun une valeur en pour-mille qui augmente en étant près du sol. Ici encore,la couleur du sol modifie la perception des senseurs et fait chuter la valeur lorsqu'il est noir et parfois aussi lorsqu'il est vert. Les valeurs retournées s'étendent environ de 275 à 965, ce qui nous force à les recalibrer.

### 1.2.2 Disposition

Les 3 capteurs sont disposé en ligne, à l'avant sous le robot. Cela permet non seulement de détecter un vide ou un sol noir, mais également de déterminer de quel côté il se trouve. Ainsi, si un seul capteur de côté perd de la valeur, on pourra influencer le robot en diminuant la vitesse de la roue opposée par exemple. Grâce au capteur central, on peut aussi déterminer avec quel angle on aborde un sol noir selon la vitesse du robot, ou influer sur l'inertie de son comportement.
(graphique : )

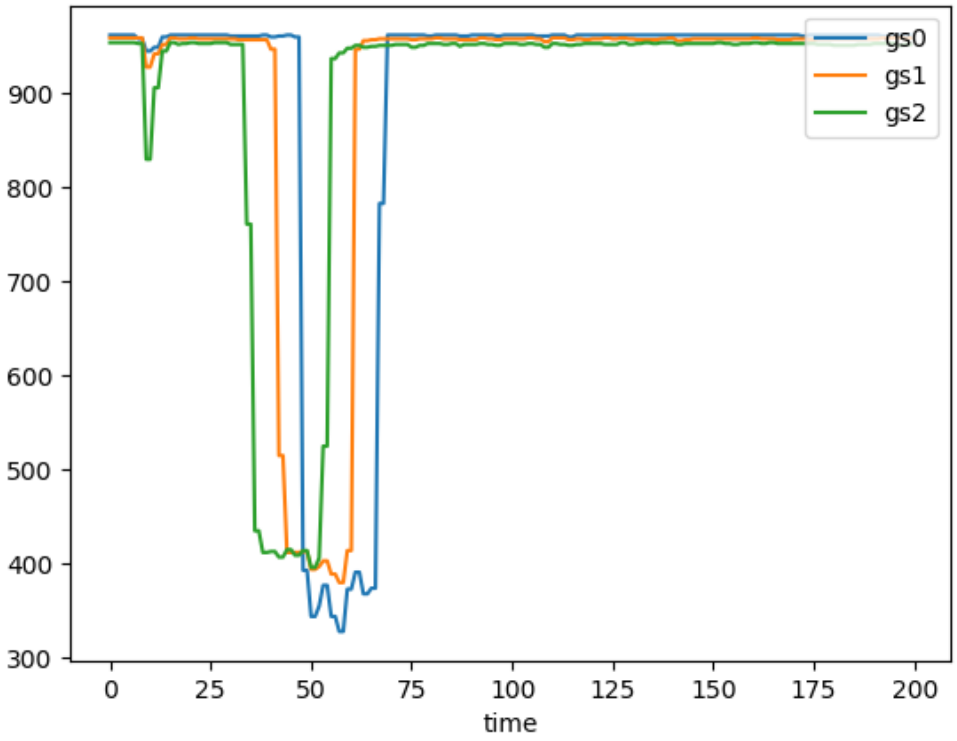


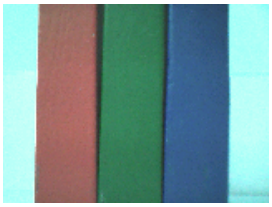

Figure 1.1 – Traversement d'une ligne noire

## 1.3 Caméra

L'e-puck 2 est équipé d'une caméra CMOS de format VGA, retournant des valeurs RGB (rouge, vert, bleu). En enregistrant des images, nous avons remarqué que la perception de la caméra est bonne au centre mais s'assombrit dans une direction radiale. Nous avons donc voulu tester sa capacité de rendu général en plaçant le robot devant des objets peints en blanc, rouge, vert ou bleu dans une salle bien éclairée.
Grâce à un graphe répertoriant les valeurs des pixels, notre première constation fût que la caméra détectait bien le blanc, avec un pic de valeurs RGB à 250 (sur 255). Pour cela, il faut néanmoins diriger la caméra vers un objet blanc lorsque l'on allume le robot, sous peine qu'il ajuste le contraste et la teinte. En effectuant les test sur plusieurs robots, nous avons réalisé que leurs différences étaient probablement dues à cette initialisation.

(a) initialisation normale (voire sous la lumière)

(b) devant un objet bleu



(c) devant un objet rouge

(d) dans le noir

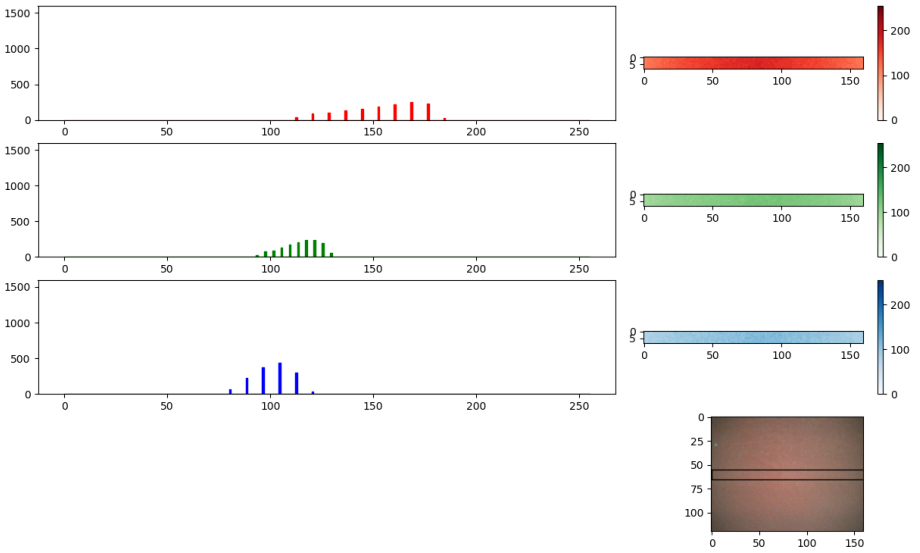



Figure 1.2 – Objet rouge, sommes des pixels par valeur, par chaîne.

Notre seconde constatation fût que les objets rouges sont mieux détectés que les verts et les bleus. En effet un objet de couleur rouge possède une moyenne de valeurs plus importante dans la chaîne des rouges que dans les deux autres. Pour comprendre cela, il faut savoir qu'un objet d'une couleur "pure" retourne en moyenne la valeur 150 dans sa propre chaîne et environ 100 dans les autres, ce qui la rend déjà différenciable. (figure1.2) Dans le cas d'un objet rouge, nous obtenons aussi ces données, mais la variance des rouges est plus importante que pour le vert ou le bleu. Nous pouvons en déduire que chaque pixel est plus vite influencé par le rouge que par les autres couleurs pour une même intensité lumineuse.

# Chapitre 2

# Comportements

Chapitre sur les comportements qui seront implémentés pour les séries.

## 2.1 Braitenberg

Les controlleurs de Braitenberg sont des programmes permettant de faire bouger un robot. Selon leur proximité avec un obstacle, il réagissent et forcent le robot à se rapprocher ou à s'éloigner. Lorsque qu'un objet et proche d'un côté, les capteurs autour du robot inhibent une roue, au choix selon le controlleur.

### 2.1.1 LOVER

Avec le Lover, le robot inhibe la roue du côté du capteur. Le robot va donc se rapprocher de l'obstacle.

### 2.1.2 EXPLORER

Avec l'Explorer, le robot inhibe la roue du côté opposé au capteu., Le robot va donc s'éloigner de l'obstacle.

## 2.2 Line-following

La couleur du sol pouvant influencer les senseurs, il est possible de détecter une épaisse ligne noire au sol. Le line-following consiste à suivre cette ligne.

### 2.2.1 Comportement de base

Pour effectuer cela, nous nous basons sur la différence des valeurs retournées par les capteurs. Si la valeur d'un capteur de côté chute, le robot est attiré et fait chuter la vitesse de la roue opposée au capteur. Les capteurs de coté permettent au robot d'être attiré par la ligne ou repoussé par les bords de celle-ci en cas de débordement. Chaque capteur prend également légèrement en compte la valeur du côté opposé. Le capteur central sert à régler l'inertie dans la rotation du robot. Nous avons donc choisi de multiplier la valeur des deux capteurs de côté par 10, de leur opposés respectifs par 2, et du capteur central par 2. Si l'on dépasse la valeur opposée avec la centrale, on obtient des mouvements saccadés.
Exemple en vidéo : l'octogone [site:youtube.com]

### 2.2.2 Dépassement frontal

Ce comportement ce base sur la différence de valeur entre les capteurs. En cas de dépassement simultané, il faut donc démarrer un compteur qui permet d'arrêter le robot et de chercher la ligne.
Autre exemple : le carré (2 sens) [site:youtube.com]

## 2.3 Wall-following

Dans le wall-following, le robot suit les murs d'une arène rectangulaire. Le robot démarre en mode Lover. Dès qu'un obstacle est rencontré, il considère qu'il s'agit d'un mur de l'arène et doit suivre les murs continuellement. La difficulté est ici d'implémenter un controlleur PID, qui décidera dans quelle mesure le robot doit s'approcher ou s'éoigner du mur.

## 2.4 Color recognition

Le "Color recognition" consiste à faire reconnaître les couleurs RGB par notre robot grâce à la caméra. La règle est simple : si le robot est face à un objet rouge, la LED sur sa gauche doit s'allumer. Il en est de même pour le vert, à l'arrière et le bleu, à droite.Vidéo [site:youtube.com]

### 2.4.1 Détection

La détection des couleurs RGB peut se faire en effectuant la somme de tous les pixels de chaque chaîne et en comparant ces trois sommes. Il n'est pas nécessaire de diviser par le nombre de pixels pour obtenir la moyenne.
Dans un premier temps, nous utilisons une fonction récupérant tous les sous-pixels de couleur séparément pour chaque chaîne, puis nous les additionnons.(figure 2.1)

```
for (int n=0; n<CAMERA_HEIGHT; n++) {
          for (int m=0; m<CAMERA_WIDTH; m++) {
              int redpixel = (int) red[n * CAMERA_WIDTH + m];
              int greenpixel = (int) green[n * CAMERA_WIDTH + m];
              int bluepixel = (int) blue[n * CAMERA_WIDTH + m];
              // printf("R:%d G:%d B:%d\n",red[0],green[0],blue[0]);
              red_count += redpixel;
              green_count += greenpixel;
              blue_count += bluepixel;
          }
      }
```

Figure 2.1 – Création des chaînes colorimétriques

Si les sommes des trois chaînes sont plus ou moins équivalentes, l'ensemble de l'image est un niveau de gris. Cependant, si l'on observe qu'une seule somme est plus prononcée que les deux autres, la couleur de l'image tend vers la chaîne R G ou B correspondante. (figure 2.2)
Le calcul de cette comparaison est le suivant : si le double de la valeur d'une chaîne surpasse celle des deux autres, on considère que la couleur correspondante a été détectée. Pour des raisons de stabilité, un ratio de $\frac{17}{16}$ multiplie les deux autres chaînes. Ainsi nous créons une zone neutre pour les gris.
Cette méthode de détection a l'avantage de fonctionner peu importe la luminosité de l'environnement. Il subsiste néanmoins un défaut : malgré la qualité de l'éclairage blanc appliqué lors de l'expérience, la somme des pixels verts tend à être plus importante que pour les autres chaînes. Il a donc fallu normaliser cette somme en la multipliant par une proportion d'environ 97%.

```
if ( 2*red_count > ((green_count+blue_count)*threshold_ratio) ) {
        enable_led(3); disable_led(2); disable_led(1);
        printf("red\n");
    }
```

Figure 2.2 – Comparaison de la chaîne rouge avec les deux autres

### 2.4.2 Implémentation en mouvement

Pour détecter les objets colorés, il faut au préalable définir la zone en contenant et les zones à ne pas observer. L'epuck n'évoluant que sur la table, nous ne prendrons en compte qu'une partie de l'image pour la détection : une ligne d'horizon de $\frac{1}{10}$ de la hauteur suffit, y compris à distance. (figure 2.3)

```
int custom_width = CAMERA_WIDTH; int custom_height = CAMERA_HEIGHT*0.1f;
for (int n=(int)((((float)CAMERA_HEIGHT)-custom_height)/2); n<(int)((((float)CAMERA_HEIGHT)+custom_height)/2); n++) {
```

Figure 2.3 – Parcours sur chaque ligne de pixel, dans l'intervalle de $\frac{1}{10}$ sous et sur la ligne d'horizon.

Le fait d'avoir mis le robot en mouvement nous a forcé à définir un seuil de détection optimal, en fonction de la distance séparant le robot et l'objet. En tentant d'augmenter cette distance, on perçoit toujours plus la facilité de détection du rouge, même après normalisation des couleurs. Le rouge peut donc être découvert beaucoup plus loin que des couleurs froides, dont la fréquence électromagnétique est plus basse.Vidéo [site:youtube.com]

## 2.5 Multi-robot coordination

### 2.5.1 Communication : généralités

L'e-puck peut communiquer virtuellement avec les autres robots par l'intermédiaire de l'ordinateur. Il peut envoyer un message vers une queue dans la mémoire tampon de l'ordinateur ou extraire le plus ancien message de cette mémoire. Bien que ce système soit fiable, nous avons remarqué deux inconvénients. Premièrement, la taille des messages est limitée à quatre caractères. Le fait de dépasser ce seuil conduit à un débordement de la mémoire. Deuxièmement, la queue enregistre les quatre derniers messages. Cela signifie qu'un robot peut envoyer plus d'un message sans causer d'écrasement, avant que le deuxième robot ne le reçoive. Cependant, le receveur ne peut pas attester de la récence de l'information reçue. Pour palier à cela, nos controlleurs possèdent un compteur qui évacue la queue avant de valider une action liée à un message reçu.

### 2.5.2 Alternative search

Le but de ce controlleur est de faire rouler deux e-puck en alternance, en implémentant deux fois le même programme. En mouvement, le robot doit démarrer dans l'état Explorer, basculer en Lover et s'arrêter dès qu'un obstacle est rencontré. (voir fig. 2.4) Le second puck démarre ensuite et reproduit le même comportement. Notez qu'en s'arrêtant,le premier puck envoie un seul message vers la queue du second. Vidéo [site:youtube.com]

Nous avons rapidement pu implémenter les trois états et les conditions de transition. Il a fallu toutefois ajuster le seuil de proximité arrêtant le robot, car lorsque deux e-pucks se rencontrent, il peinent à se détecter. En effet, un léger espace existe lorsque les robots se touchent, et

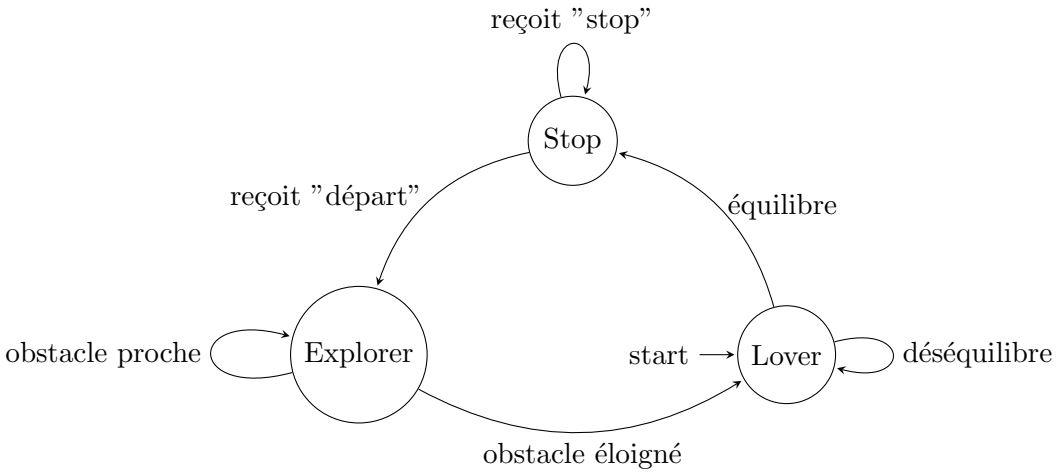



Figure 2.4 – FSM diagram in LaTeX.

leur composition en plastique reflète mal les infrarouges. Une difficulté plus importante s'est révélée ensuite : lorsqu'un robot s'arrête, l'autre doit repartir. Or il est arrivé que les deux restent à l'arrêt ou repartent en même temps. Le problème est lié à aux messages restants dans la queue : si un robot écoute un message désuet, il réagira potentiellement de manière inappropriée. En écoutant un message extraordinaire (changement d'état), il convient donc de nettoyer tous les autres messages reçus entre temps. D'autres tactiques se révèlent inefficaces, comme la communication permanente ou la rupture de l'écoute lors d'un changement d'état.
Nous avons ensuite testé l'Alternative Search sur trois robots. Deux robots tendent à partir en même temps et font l'inverse du troisième. Le résultat est cependant relativement brouillon car un robot qui s'arrête envoie une notification de départ. Si un robot est déjà en marche, il va l'enregistrer pour l'écouter plus tard. Une idée pour résoudre cela serait d'évacuer la queue au début de l'écoute ou d'enregistrer les états des pucks selon leur identifiant. Vidéo [site:youtube.com]
le résultat produit sur 3 pucks est aussi intéressant sans évacuer la queue. Si celle-ci contient une notification de départ, elle fonctionne alors comme un compteur selon la position du message dedans. Il s'ensuit que les 3 pucks peuvent s'arrêter et partir à tour de rôle, toujours dans le même ordre.

### 2.5.3 Simultaneous search

Avec ce controlleur, 3 robots roulent en Lover et s'arrêtent s'ils rencontrent un obstacle. Dès que tous sont 5arrêtés, ils repartent simultanément en Explorer, puis re-basculent en Lover.
Pour implémenter cela, nous reprenons l'Alternative Search et changeons la condition pour le démarrage. Un robot devra attendre le signal des deux autres robots avant de démarrer lui-même. On notera que cette méthode nécessite d'écouter les autres robots en permanence, car un robot peut s'arrêter et envoyer un message pendant que le robot destinataire est en marche.Vidéo [site:youtube.com]

# Chapitre 3

# Conclusion

Cette expérience nous aura permis d'acquérir une vision globale des difficultés liées à de vrai robots. Nous avons pu constater que des différences existent entre un simulateur et un robot. De plus, on trouve des différences liées a des paramètres variables, et des différences existent parfois d'un robot à l'autre. Pour poursuivre cette expérience, il serait intéressant de développer la communication entre les robots afin de pallier à ces paramètres.

# Bibliographie


[1] Justin Zobel. *Writing for Computer Science*, 2nd edition. Springer-Verlag, London, 2004, 275 pages.

[2] Valentino Braitenberg. *Vehicles : Experiments in Synthetic Psychology*. MIT Press, 1986.

[3] *Webots Reference Manual*. `https://www.cyberbotics.com/reference.pdf` version 2019a Last visited : 11.02.2019.

# Appendix

## Appendix A Experimental Results

Place to list the gathered data.

## Appendix B Source Code

Place to list source code.

### B.1 IR sensors calibration procedure

The code below shows the IR sensor calibration procedure.

```
// get the correction values for prox sensors
void get_prox_corr_vals() {
  int i, j;

  // init array for calibration values
  for (i=0; i<PROX_SENSORS_NUMBER; i++) {
    prox_corr_vals[i] = 0;
  }

  // get multiple readings for each sensor
  for (j=0; j<NBR_CALIB && wb_robot_step(TIME_STEP)!=-1; j++) {
    for (i=0; i<PROX_SENSORS_NUMBER; i++) {
      prox_corr_vals[i] += wb_distance_sensor_get_value(prox_sensor_tags[i]);
    }
  }

  // calculate average for each sensor
  for (i=0; i<PROX_SENSORS_NUMBER; i++) {
    prox_corr_vals[i] = prox_corr_vals[i] / NBR_CALIB;
  }
}
```

## Appendix C LaTeX Examples

This section shows some common uses of LaTeX features.

### C.1 Images

Example of how to include an image can be seen in Figure 3.1. All figures must be referenced somewhere in the report.

### C.2 Tables

Example of how to include a table can be seen in Figure 3.2. All figures must be referenced somewhere in the report.

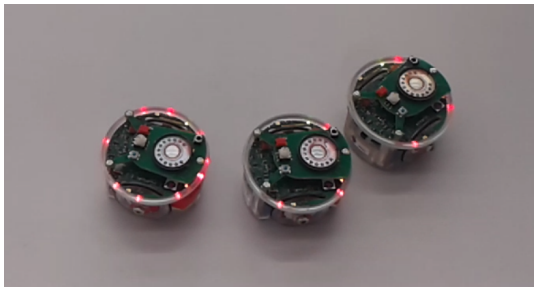


Figure 3.1 – Including an image.

| Title 1 | Title 2 |
|---|---|
| item 11 | item 12 |
| item 21 | item 22 |

Figure 3.2 – Table with caption.

## C.3 Listings

Example of how to include listing can be seen in Figure 3.3 and Figure 3.4. All figures must be referenced somewhere in the report.

```
// get the correction values for prox sensors
void get_prox_corr_vals() {
  int i, j;

  // init array for calibration values
  for (i=0; i<PROX_SENSORS_NUMBER; i++) {
    prox_corr_vals[i] = 0;
  }

  // get multiple readings for each sensor
  for (j=0; j<NBR_CALIB && wb_robot_step(TIME_STEP)!=-1; j++) {
    for (i=0; i<PROX_SENSORS_NUMBER; i++) {
      prox_corr_vals[i] += wb_distance_sensor_get_value(prox_sensor_tags[i]);
    }
  }

  // calculate average for each sensor
  for (i=0; i<PROX_SENSORS_NUMBER; i++) {
    prox_corr_vals[i] = prox_corr_vals[i] / NBR_CALIB;
  }
}
```

Figure 3.3 – Listing included from file.

```
// constrain speed to +/- MAX_SPEED
double bounded_speed(double speed) {
  if (speed > MAX_SPEED) return MAX_SPEED;
  else if (speed < -MAX_SPEED) return -MAX_SPEED;
  else return speed;
}
```

Figure 3.4 – Listing within LaTeX.

## C.4 Font Style and Text Size

The font style may be modified : **bold**, *italic*, *Emphasis*, Capitals, `verbatim`, etc.
The text size can be changed : tiny, small, large, huge, etc.

## C.5 Enumerations and Other Lists

Enumerations are easy, there is the `enumerate` environment :

1. First item
2. Second item
3. Third item

For lists, there is the `itemize` environment :

— First item
— Second item
— Third item

For definitions lists, there is the `description` environment :

**First term** – Description of the first term

**Second term** – Description of the second term

## C.6 Quotations and References

Books and other documentation can be referenced as [2] and websites as [3].

## C.7 FSM diagram

In Figure 3.5 is depicted a Finite State Automata diagram as presented in Lecture 02.

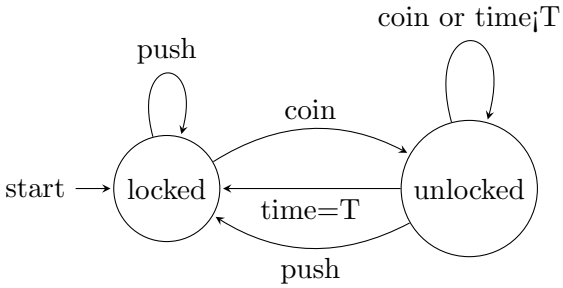



Figure 3.5 – FSM diagram in LaTeX.