Robotics Project I

Firstname Lastname, Group 0

IN.2022 Robotics 2019, BSc Course, 2nd Sem.
University of Fribourg
firstname.lastname@unifr.ch

Abstract

Brief description of the content (5-10 lines). Helps people decide whether the report is relevant for them or not. Usually written at the end.

Keywords: add, keywords, for, indexing

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]

Contents

Chapter 1

Introduction

Objectives of this project, and brief description of the structure of the report.

Chapter 2

Sensors

Chapter about the sensors that will be used during Project I

2.1 Proximity infra-red sensors

In this section, the only sensors that are concerned are the front right and front left infra-red sensors (respectively "ps0" and "ps7" in Figure 2.1). To test them, the robot number 204 is firstly placed on a table, free of any obstacle, to calibrate the sensors. This calibration phase is necessary to suppress some amount of possible noise on the sensors on real bots. It is then placed in front of an obstacle and it is commanded to move away at a determined speed while taking periodic measurements. The data collected is put in a csv file and plotted with the help of a python script. The result can be seen in Figure 2.1.

The y-axis represents the proximity from the object, the higher the number the lower the luminosity, meaning the closer the obstacle. The x-axis represents the steps, it acts as a time notion.
Compared to the reference graph in Figure 2.2 given as part of this exercise, the curve follows approximately the same pattern after a certain number of steps, if not for some perturbation. The fact that the curve follows a linear pattern in the first steps is due to the robot taking some measurements before moving away, thus staying stationary in those steps. The perturbations are probably due to dust reflecting the light, glitch, unpredicted movement of the support or other form of noise.

2.2 Infra-red ground sensor

The goal here is to test the infra-red ground sensors. The robot is equiped with three sensors underneath it that work in the same way as the proximity sensors. To test them, the robot number 206 crosses a thick black line in two different manner: perpendicularly and diagonally. The data is collected and put in a csv file which is then plotted using a python script.

There is three graphs, one for each ground sensor. As with the proximity sensors' response graphs, the x-axis represents the steps, which act as a time notion. The y-axis here is inverse to the proximity sensors, the lower the value the lower the luminosity. As we can see in Figure 2.3, representing the perpendicular crossing, the three sensors detect the line at approximately the same time and in Figure 2.4, representing the diagonal crossing, the sensors are lightly delayed. This let us understand that the three sensors are aligned in a perpendicular manner below the

robot.

2.3 Camera

Description of the camera and graphs of measurements

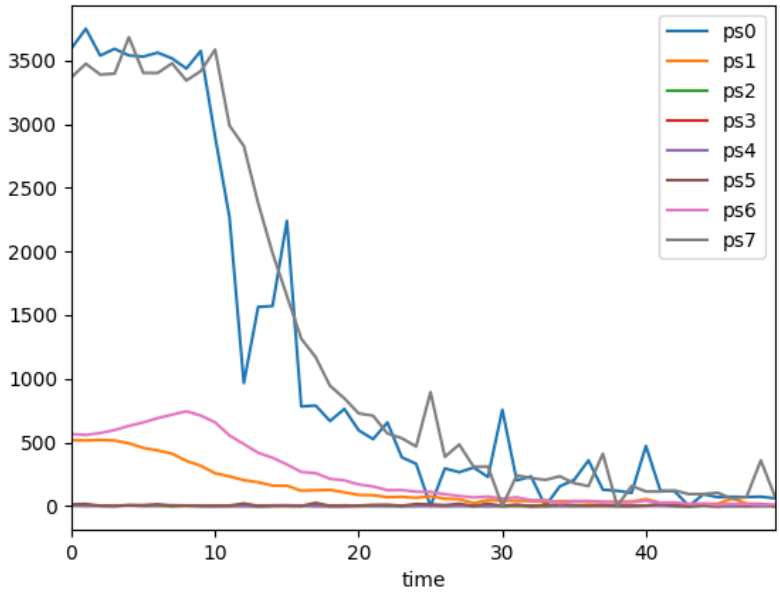

Figure 2.1: The calibrated measurement of robot 204

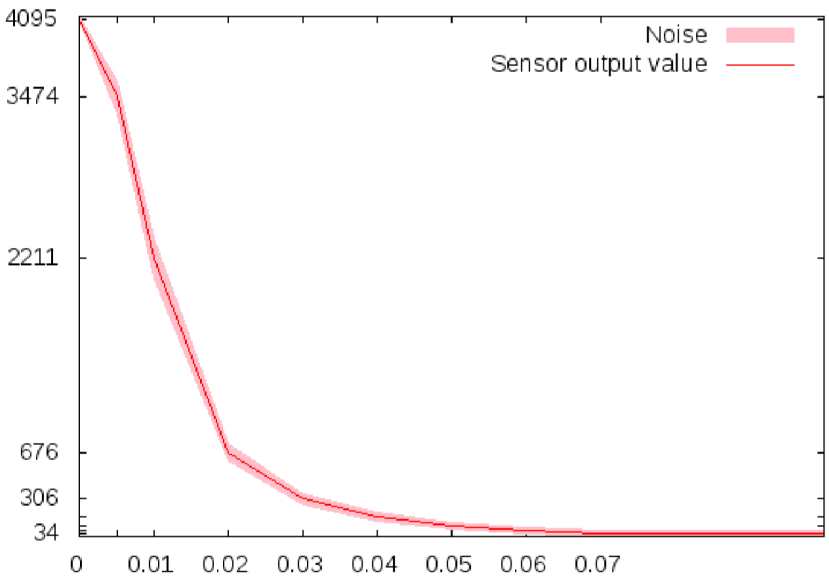

Figure 2.2: The reference graph

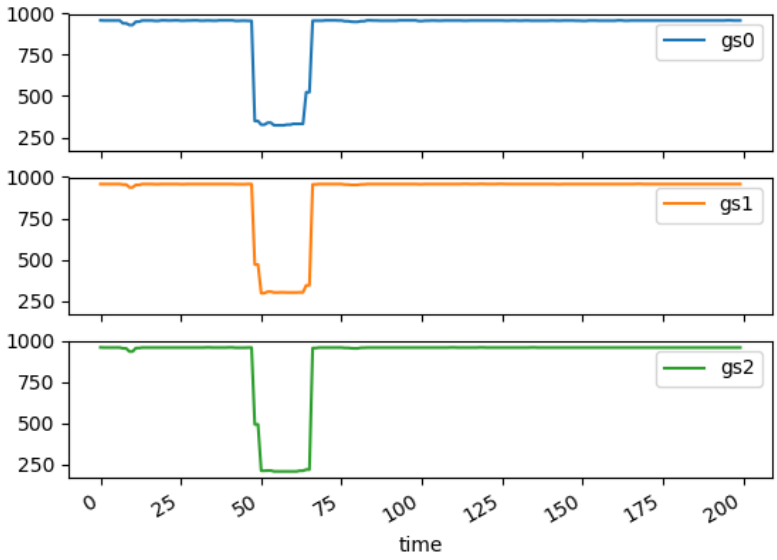

Figure 2.3: Ground sensor response graph perpendicular

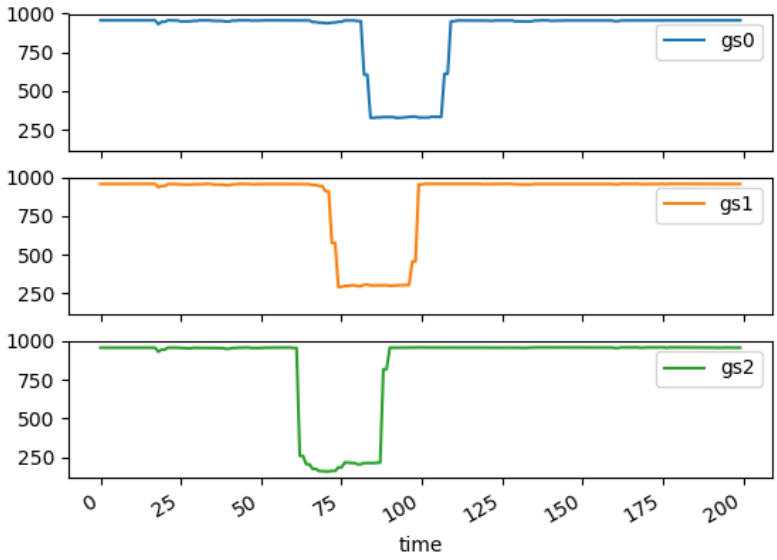

Figure 2.4: Ground sensor response graph diagonal

Chapter 3

Behaviours

Chapter about the behaviours that will be implemented for the assignments

3.1 Braitenberg vehicle

The Braitenberg vehicle is a thought experiment of an abstract vehicle equipped with motors and sensors said to mimic minimalist animal behaviours. The simplest Braitenberg vehicle is comprised of a motor linked to a sensor. The more the sensor is stimulated, the slower the motor performs.
By combining two motor and two sensors, it is possible to create more complex behaviours as seen in Figure 3.1. Two of those behaviours called "lover" and "explorer" are described in more details in section 3.1.1 and 3.1.2. We are trying, in this section, to reproduce those behaviours with the help of the e-pucks.

3.1.1 LOVER

The lover behaviour is composed of two motors each linked to a sensor and a wheel, one on the left and one on the right, in the same manner as in point a of Figure 3.1. Since the motors slow down the closer the sensors gets to the stimulus, the vehicle gets closer to the stimulus. In our experience, the stimuli are represented as obstacles.
The proximity formula for the e-puck is calculated as the average of the four front sensors.
The speed of the wheels are reduced proportionally to its proximity formula. If the robot goes

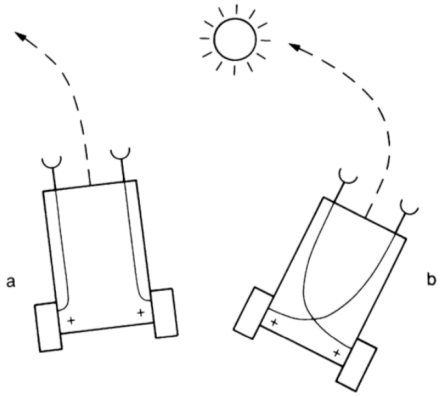

Figure 3.1: Different behaviours

too close to an obstacle, its speed ends up being negative and it goes backward for a short time until it is far enough for the speed to become positive again.

3.1.2 EXPLORER

The explorer behaviour follow a similar structure as the lover behaviour except the sensor are linked to the opposite motors. Therefore, the vehicle behave in the opposite way of the lover behaviour and will avoid any stimuli.
Two proximity formulas, one for the left wheel and one for the right wheel, are calculated identically as in the lover behaviour.However, they are both weighted, in the same manner, with the two frontal sensors (prox7 and prox6 for the left, prox1 and prox0 for the right) without weights, the side one (prox5 for the left, prox2 for the right) weighted at 70% and the one on the back (prox4 for the left, prox3 for the right) weighted at 20%.
The speed of the wheel is set in the same manner as the lover mode, the difference being the each wheel is set separately since the proximity formula is not the same for each one. The speed of each wheel is also reduced by 550 units to avoid collision with the obstacle.

3.1.3 Exploring Lover

The Exploring Lover behavior is a combination of both the EXPLORER and LOVER behavior which allow the robot to approach obstacles and avoid them by switching between the two. The robot switches behavior whenever it reaches an equilibrium point. In LOVER state, the equilibrium would be that the robot reached an obstacle, as in EXPLORER, it would be that the robot no longer perceive any obstacle in front of it. For a better understanding, refer to the Figure 3.2 which features the finite-state machine diagram of this behaviour.
To find the equilibrium point in LOVER state, a counter is set. Each time the robot goes back and forth the counter is incremented. If the counter reaches 10, the robot is considered to have found the equilibrium point.
In EXPLORER state, the equilibrium is reached whenever the average of the front and side proximity sensor are below 50 unit, meaning there is no longer any obstacle in the e-puck's path.

Here is a video of the robot in action: `https://youtu.be/rDlxz8RVZrM`

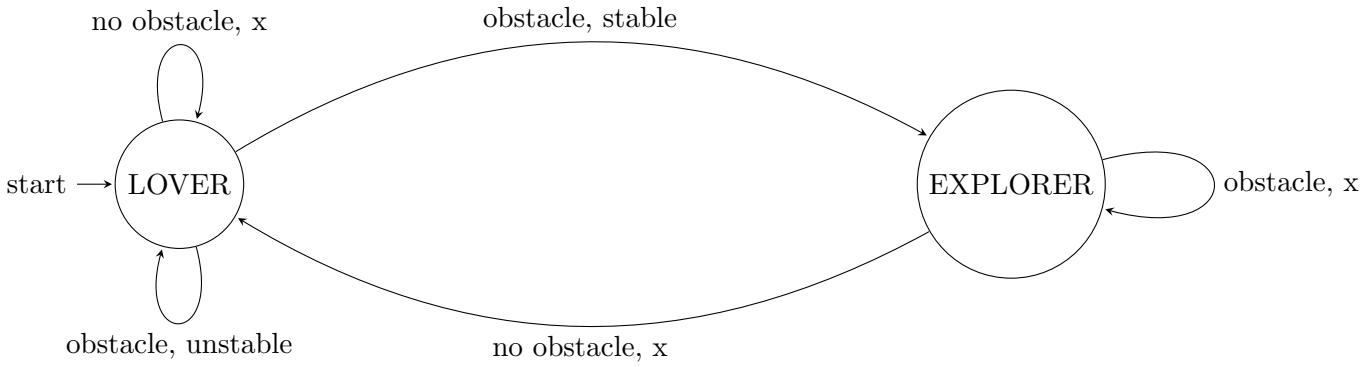

Figure 3.2: The FSM schema of the Exploring Lover Behaviour

3.2 Line-following

For this experience, the robot's goal is to follow a thick black line drawn on a sheet of hard paper. Two shapes are to be followed, a square and an octagon. To do so, the robot uses two states, a state for looking for the line and a state for following the line. In searching state, the robot first turn on right for a certain amount of time determined by a counter as seen in Figure 3.3. It is an arbitrary set value that allow it to seemingly rotate for 90 degrees, after that the robot turns on right for an amount of approximately 180 degrees also determined by an arbitrary counter and finally moves forward if it has not found any line. Whenever the central ground sensor detects a line or more specifically when it returns a value lower than 600, the robot switches to line following mode. In following state, the left and right wheels speed is bound to their respective ground sensor, so when the left or right ground sensor are off the line their respective wheel accelerate in order to keep the robot on it, as shown in Figure 3.4. The initial rotation of the robot in searching mode might seem weird but it is indeed used in order for the robot to follow sharp turns as in the rectangle line, without this rotation the robot would move out of the shape whenever it would enc outer a sharp turn since its center ground sensor would be off the line.
The result can be seen here: `https://www.youtube.com/watch?v=MpUPNzAhKQQ`

3.3 Wall-following

The wall following is done using a proportional–integral–derivative controller or PID controller which is a control loop feedback mechanism. The PID controller is given a set point, calculates the error between this set point and the measured point and applies correction based on proportional, integral and derivative terms, hence the name.

There is three parameters of the PID controller that are modified in order to make the robot follow a wall in a decent way. Those three parameters are K for the proportional term, T_I foe the integral term and T_D for the derivative term. Those parameters are set in a trial and error manner in order to obtain the best possible outcome. In addition to the PID, the robot is able to switch between right and left mode depending on its orientation when approaching an obstacle. The final values that are used in the experience can be seen in Figure 3.5 and a video demonstrat-

```
1  void search_line(){
3      get_ground(IR_ground);
5      if(IR_ground[GS_CENTER] > 600){
7          switch(state){
9              case 0:
11                 if(counter < 20){
13                     set_speed(-NORM_SPEED,NORM_SPEED);
                       counter++;
15
                   } else{
17
                       counter = 0;
19                     state = 1;
21                 }
                   break;
```

Figure 3.3: The searching mode for the following line controller

ing the final behaviour is available here: `https://www.youtube.com/watch?v=H8ZilBo4E1c`

Once the robot follows the wall smoothly, the PID parameters can be observed. Any changes in either of the three parameters of the controller, should modify the behaviour of the robot in a certain way. To observe the changes, the robot is asked to follow a wall for certain time with the initial values first as a reference. It is placed in front of the wall, moves forward, turns either left or right depending on its initial orientation and follows the wall for a while until it reaches a corner and turns to follow it. The reference graph with the initial values can be seen in Figure 3.6. The changes that interest us the most are the one applied on the value ds which is the speed difference applied to the wheels after correction from the PID controller.
The first parameter to be modified is the proportional term K. It is increased from 1.5 to 5 and the results can be observed in Figure 3.7. The graph shows that ds seem to vary a lot even when the robot is going straight. From that we suppose that K controls the "smoothness" of the movement.
The second parameter is the derivative term T_D. It is increased from 10^{-5} to 0.5. As seen in Figure 3.8, T_D seem to have an effect on rotations. On each rotation, ds seem to switch between positive and negative value a lot, making rotations seem more hesitant.
The last parameter is the integral term T_I. It is decreased from 800 to 50 and the result is shown in Figure 3.9. It doesn't seem to have much of an effect on the ds. It is hard to extract anything from this.

3.4 Color recognition

Description of the color recognition behaviour

3.5 Multi-robot coordination

Description of the Multi-robot coordination using communication between robots

```
//If white is detected on the left sensor, the robot shifts on the right, and conversely.

void follow_line{}{

    //...//

    double gs_left = (1 * IR_ground[GS_LEFT]) / 1.0;
    double gs_right = (1 * IR_ground[GS_RIGHT]) / 1.0;

    double ds_left = (NORM_SPEED * gs_left) / 900;
    double ds_right = (NORM_SPEED * gs_right) / 900;

    double speed_left = NORM_SPEED - ds_right;
    double speed_right = NORM_SPEED - ds_left;

    speed_left = bounded_speed(speed_left);
    speed_right = bounded_speed(speed_right);

    set_speed(speed_left, speed_right);

}
```

Figure 3.4: Wheels speed settings for the follow line behaviour

```
#define K 1.5
#define T_I 800
#define T_D 0.00001
```

Figure 3.5: The final PID controller values

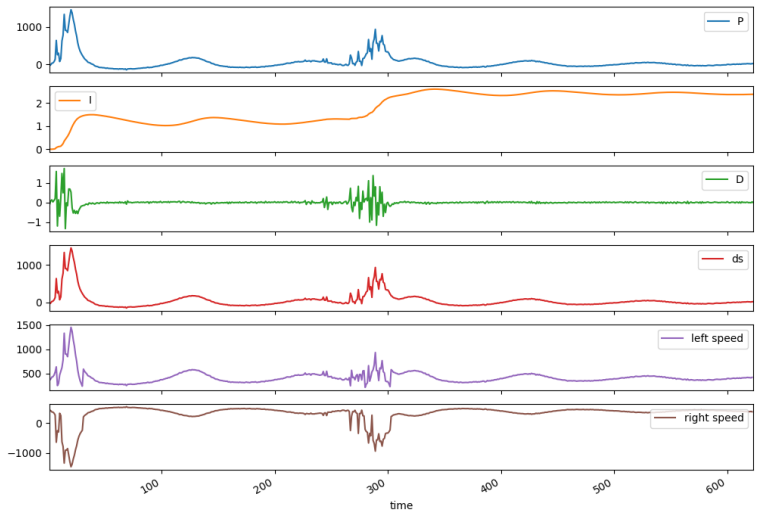

Figure 3.6: The reference base graph without any value changes

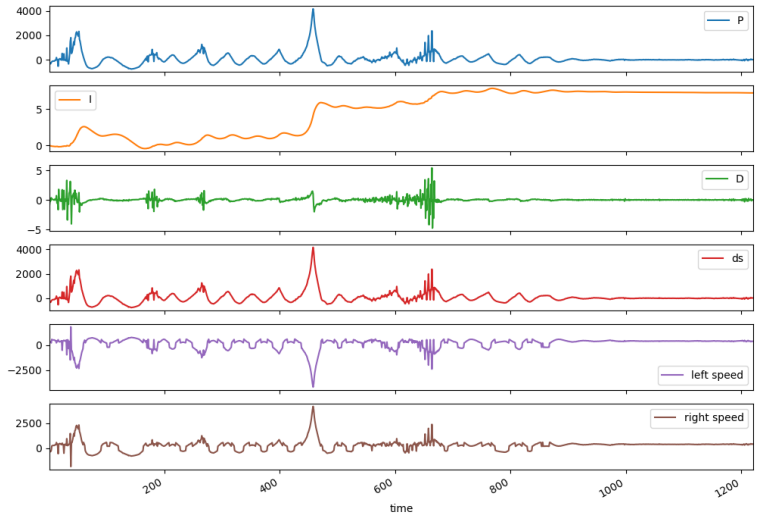

Figure 3.7: PID controller with variable K=5

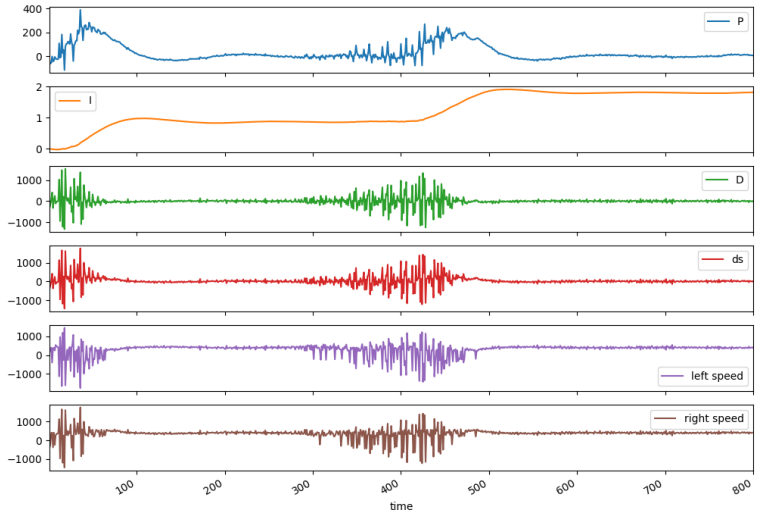

Figure 3.8: PID controller with variable T_D=0.5

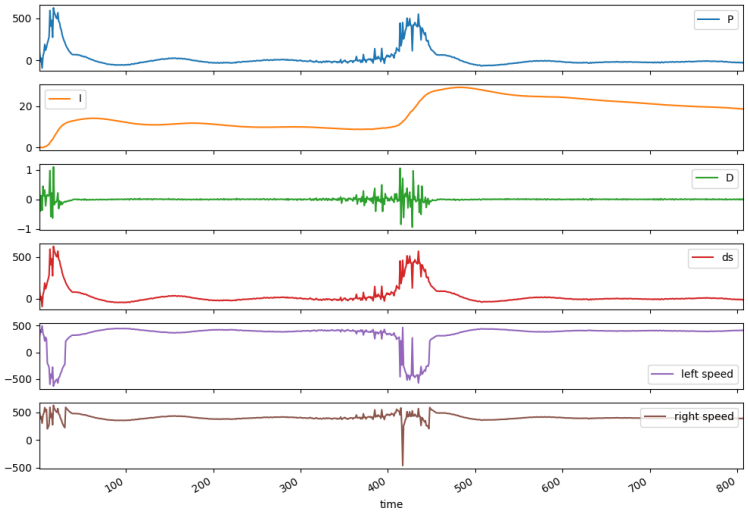

Figure 3.9: PID controller with variable T_I=50

Chapter 4

Conclusion

Synthesis of the report and outlook for further work.

Bibliography

[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 4.1. All figures must be referenced somewhere in the report.

C.2 Tables

Example of how to include a table can be seen in Figure 4.2. All figures must be referenced somewhere in the report.

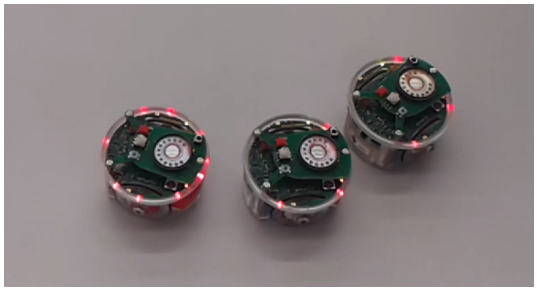

Figure 4.1: Including an image.

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

Figure 4.2: Table with caption.

C.3 Listings

Example of how to include listing can be seen in Figure 4.3 and Figure 4.4. 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 4.3: 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 4.4: 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 4.5 is depicted a Finite State Automata diagram as presented in Lecture 02.

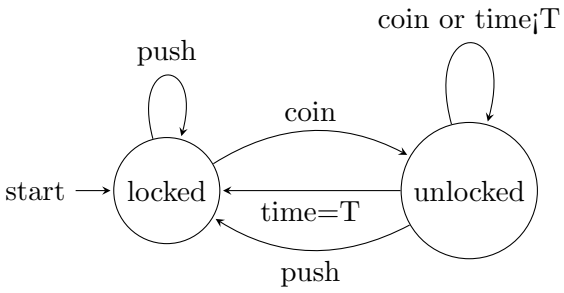

Figure 4.5: FSM diagram in LaTeX.