

Lab 2: Control

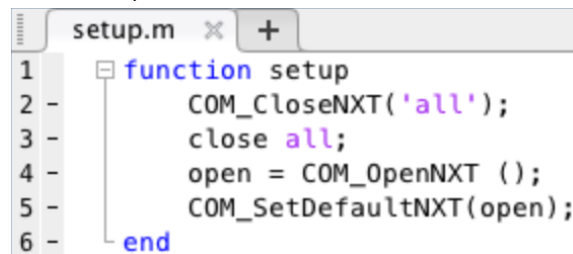
Sensors: Whenever we use to write `SENSOR_2` in our code, we mean the left sensor (if one looks at the front of the robot and with front we mean the side were the ultrasonic sound is send out and where it also is (received). Hence, when we write `SENSOR_3`, we mean the right sensor (if one looks at the front of the robot).

Setup:

For each exercise we always needed the following code:

```
COM_CloseNXT('all');  
close all;  
open = COM_OpenNXT ();  
COM_SetDefaultNXT(open);
```

So, we decided to not directly include this in our code. Instead, we made another file, called 'setup.m' and this file contains a function called 'setup' (in the code: `function setup`). The function `setup` prepares the components we need in each of our codes (for exercises 1, 2a, 2b, 2c, and 3). Here is our code:



```
1 function setup  
2     COM_CloseNXT('all');  
3     close all;  
4     open = COM_OpenNXT ();  
5     COM_SetDefaultNXT(open);  
6 end
```

In order to use the above described code, we use the following command before we run our MATLAB code for exercises 1, 2a, 2b, 2c, and 3:

```
>> setup Filename;
```

For exercise 1, we use this command:

```
>> setup Exercise1;
```

For exercise 2a, we use this command:

```
>> setup Exercise2a;
```

For exercise 2b, we use this command:

```
>> setup Exercise2b;
```

For exercise 2c, we use this command:

```
>> setup Exercise2c;
```

For exercise 3, we use this command:

```
>> setup Exercise3;
```

Calibration:

We calibrated the robot in order to determine what value should be considered as black and what value should be considered as white. To calibrate the robot, we first put the robot on the black tape. The light sensor measured the value of the black tape. We displayed this value in

MATLAB, so that we could determine a threshold (so, what range we considered as black and what range we considered as white). We did the same to measure the white surface (to get the value of the white surface). Now, we will describe the code we used and what we did in order to obtain the previously described values.

First, we wrote the code for the calibration and we put the code into a file called 'Calibration.m'. This is our code:

```
setup.m x Calibration.m x +
1 % First, we open one of the light sensors:
2 - OpenLight(SENSOR_2, 'ACTIVE')
3 % Then, we place the robot on the white surface.
4 % And, we get the value that is reflected by the white surface.
5 % We assign this obtained value to the variable val_white:
6 - val_white = GetLight(SENSOR_2);
7 % We want to know the value, so we the value:
8 - disp(val_white);
9 % This time, we place robot in such a way that
10 % SENSOR_2 detects the value reflected by the black tape.
11 % We do the same to obtain the value for the black tape:
12 - val_black = GetLight(SENSOR_2);
13 % We also want to know this value, so we display this value too:
14 - disp(val_black);
15 % Once we are done, we can close the light sensor:
16 - CloseSensor(SENSOR_2);
```

For the calibration, we also needed the code of the 'setup.m' file. So, when we want to calibrate the robot, we use the following command:

```
>> setup Calibration;
```

Then, we actually calibrate the robot by running the code from the file called 'Calibration.m'. We obtained these values by running the code:

```
>> Calibration
723

371
```

So, we know that the value for white is equal to 723 and thus, we consider a value of 600 or bigger as white surface (this means that the robot detects white surface) in our codes. We also know that the value for black is equal to 371 and therefore, we consider a value of 400 or smaller as black tape (this means that the robot detects black tape) in our codes.

Exercise 1: Write a MATLAB program to make the robot randomly drive inside an area marked with black tape. The robot should first drive towards the line and then continue moving, without ever crossing it.

Answer: This is our MATLAB Code:

```
% Instead of the following four lines indicated as
% line 1, 2, 3, and 4, we use the command '>> setup Exercise 3'.
```

```

% COM_CloseNXT('all');          % line 1
% close all;                    % line 2
% open = COM_OpenNXT();         % line 3
% COM_SetDefaultNXT(open);     % line 4
% So, our actual code begins here:
% The left light sensor is opened:
OpenLight (SENSOR_2,'ACTIVE');
% The right light sensor is opened:
OpenLight (SENSOR_3,'ACTIVE');
% Both sensors detect if the robot is encountering the black line.
% The sensors detect the surface and we store the obtained value
% which represents the colour (in our case: either black or white).
% We store these values in two vectors, one vector for each sensor.
% So, we also need a variable which will be the index of our vector.
% We initialize the variable for the index of our vector:
vec_index = 1;
% We use a while loop, because we want our robot to keep moving:
while (true)
    % We make an object for both motors. We set the speed of the
    % motors to 15 (which is 15% of the maximum speed):
    mAC = NXTMotor ('AC', 'Power', 15);
    % The motors make the robot move:
    mAC.SendToNXT();
    % We want to obtain values of the surface to detect if the
    % robot is encountering the black line. We get the values from
    % both sensors. We store both values in two different vectors.
    % One vector is for the left light sensor:
    vec_val_2(vec_index) = GetLight(SENSOR_2);
    % And the other vector is for the right light sensor:
    vec_val_3(vec_index) = GetLight(SENSOR_3);
    % If the left light sensors detects black:
    if vec_val_2(vec_index) < 400
        % The robot will first move backwards. We achieve this by
        % using a negative value for the speed. The robot will
        % move backwards with a speed of this value multiplied by
        % minus one (-20 * -1 = 20):
        mAC = NXTMotor('AC', 'Power', -20);
        % Now, the robot moves backwards with a speed of 20:
        mAC.SendToNXT ();
        pause(1);
        % The robot turns:
        mA = NXTMotor('A', 'Power', 15);
        mC = NXTMotor('C', 'Power', 30);
        mA.SendToNXT ();
        mC.SendToNXT ();
        pause (1);
    % We end the if-statement:

```

```

end

% If the right sensors detects black:
if vec_val_3 (vec_index) < 400
    % We do the same when the right light sensor detects the
    % black line
    mAC = NXTMotor ('AC', 'Power', -20);
    mAC.SendToNXT ();
    pause(1);
    % The robot turns:
    mC = NXTMotor ('C', 'Power', 15);
    mA = NXTMotor ('A', 'Power', 30);
    mC.SendToNXT ();
    mA.SendToNXT ();
    pause (1)
% We end the if-statement:
end
% Before the end of the while-loop, we increment our variable
% vec_index by one, so we can store the next obtained values
% from the light sensors if the the loop is executed again:
vec_index = vec_index + 1;
% This process happens every 0,1 seconds. We do this every 0,1
% seconds, because we want to detect the value of the surface
% every 0,1 seconds:
pause (0.1);
% We end the while-loop:
end

```

Exercise 2:

2a: Write a MATLAB program that makes the robot drive, and then stop in front of the first encountered (fixed) obstacle. You need to use the motor functionality in the way you used it in Example 3.1, and use an ultrasound sensor in the way you used it in Example 3.2.b). Beware that the minimal distance measured by the ultrasound sensor is not 0 cm. Use the WatchSensor GUI to measure the actual minimal value.

Answer: This is our MATLAB Code:

```

% This code makes robot drive. The robot stops in front of first
% encountered fixed obstacle. We detect the obstacle by using the
% ultrasound sensor by sending out a sound with a frequency that is
% beyond human hearing; Then, the time of flight, the time it takes
% for the sound to be reflected back, is measured. This time of
% flight (ToF) is then multiplied by the sound speed (this is a
% constant that is equal to 300 m/s) and then this value is also

```

```

% divided by two. This formula is used:  $ToF * c / 2$ ; Now, the
% distance to the object is calculated.
% Instead of the following four lines indicated as
% line 1, 2, 3, and 4, we use the command '>> setup Exercise 3'.
% COM_CloseNXT('all');           % line 1
% close all;                     % line 2
% open = COM_OpenNXT();          % line 3
% COM_SetDefaultNXT(open); % line 4
% So, our actual code begins here:
% We want our robot to move straight forward with 30% of its maximum
% speed:
mAC = NXTMotor ('AC', 'Power', 30);
% We open the ultrasonic sensor of the robot:
OpenUltrasonic (SENSOR_1);
% Once we have set the speed and opened the ultrasonic sensor, we
% make the robot move at the desired speed:
mAC.SendToNXT ();
% We determine the distance that the robot should keep to the object.
% This distance cannot be zero, because the ultrasonic sensor is
% placed not directly at the front of the robot:
minDistance = 30;
% The ultrasonic sensor will measure the distance to the object. We
% store this obtained value (the distance to the object) in a vector.
% So, we also need a variable which will be the index of our vector.
% We initialize the variable for the index of our vector:
vec_index = 1;
% We use a while-loop, because we want our robot to keep moving
% (unless the robot reaches the minimum distance to the object):
while (true)
    % The ultrasonic sensor obtains the value of the distance
    % between object and this value is stored as an element with
    % index vec_index in a vector:
    val(vec_index) = GetUltrasonic(SENSOR_1);
    % Stops if the distance is less than the minimum distance:
    if val(vec_index) < minDistance
        mAC.Stop ('off');
    % We end the if-statement:
    end
    % Before the end of the while-loop, we increment our variable
    % vec_index by one, so we can store the next obtained value
    % from the ultrasonic sensor if the the loop is executed again:
    vec_index = vec_index + 1;
    % This process happens every 0,1 seconds. We do this every 0,1
    % seconds, because we want to measure the distance to the
    % object every 0,1 seconds:
    pause (0.1);
end

```

```
% We end the while-loop:  
end
```

2b: Write a MATLAB program that instructs the robot to follow a moving object. If the object stops, the robot must also stop. If the object moves (slowly), the robot must drive straight forward and follow the obstacle. You should reuse the program from exercise 2a.

Answer: This is our MATLAB code:

```
% Instead of the following four lines indicated as  
% line 1, 2, 3, and 4, we use the command '>> setup Exercise 3'.  
% COM_CloseNXT('all');      % line 1  
% close all;                % line 2  
% open = COM_OpenNXT();     % line 3  
% COM_SetDefaultNXT(open); % line 4  
% So, our actual code begins here:  
% We want our robot to move straight forward with 20% of its maximum  
% speed:  
mAC = NXTMotor ('AC', 'Power', 20);  
% We open the ultrasonic sensor:  
OpenUltrasonic (SENSOR_1);  
% Once we have set the speed and opened the ultrasonic sensor, we  
% make the robot move at the desired speed:  
mAC.SendToNXT ();  
% We define the minimum distance to the object:  
minDistance = 25;  
% We define the maximum distance to the object. If the robot is at a  
distance of more than 100 to the object, then it will move faster.
```

```

maxDistance = 100;
% We define the average distance to object. This distance should be
kept while following the object:
averageDistance = 50;
vec_index = 1;
% We use a variable x to determine when we want the condition for
% the while-loop to be true. Thus, we use this variable to exit the
% while-loop when it is the right moment (i.e. when the object is
% standing still):
x = 1
% We use a while-loop and at the beginning the condition of the
% while-loop is true (x = 1, which means that x > 0), but it is not
% true if x becomes less than or equal to zero:
while (x > 0)
    % The ultrasonic sensor measures the distance to the object it
    % is following and this value (the distance to the object is
    % stored as an element with the index vec_index in the vector:
    val(vec_index) = GetUltrasonic (SENSOR_1);
    % If the distance to the object is greater than the maximum
    % distance (maxDistance):
    % increase speed so that the robot is not far behind
    % the object.
    if val(vec_index) > maxDistance
        % The speed of the robot is increased so that the robot is
        % not far behind the object:
        mAC = NXTMotor ('AC', 'Power', 40);
        mAC.SendToNXT ();
    % We end the if-statement:
    end
    % If the distance is less than or equal to the minimum
    % distance:
    % first check whether the object has stopped or slowed down
    if val(vec_index) <= minDistance
        % First, the robot stops
        mAC.Stop ('brake');
        % The robot stops for three seconds to make sure whether
        % the object has stopped moving or slowed down:
        pause(3);
        % The robot measures the distance to the object:
        pause_distance = GetUltrasonic (SENSOR_1);
        % If the pause_distance is the same or less than the
        % measured distance before the robot stopped, then the
        % object has stopped:
        if pause_distance <= val(vec_index)
            % So, we stop the robot from moving:
            mAC.Stop ('off')
            % By assigning the value zero to the variable x, the

```

```

        % condition for the while-loop (condition: x must be
        % bigger than zero) is false and thus, we will exit
        % the while-loop:
        x = 0;
    % We end the if-statement:
    end
% We end the if-statement:
end
% Before the end of the while-loop, we increment our variable
% vec_index by one, so we can store the next obtained value
% (the distance to the object) from the ultrasonic sensor if
% the the loop is executed again:
vec_index = vec_index + 1;
% This process happens every 0,1 seconds. We do this every 0,1
% seconds, because we want to measure the distance to the
% object every 0,1 seconds:
pause (0.1);
% We end the while-loop:
end

```


2c: Avoid an obstacle Write a MATLAB program that moves around inside an area marked with black tape, without crossing the line, but also avoids collision with any obstacle on its way and continue driving.

Answer: This is our MATLAB code:

```
% Instead of the following four lines indicated as
% line 1, 2, 3, and 4, we use the command '>> setup Exercise 3'.
% COM_CloseNXT('all');      % line 1
% close all;                % line 2
% open = COM_OpenNXT();     % line 3
% COM_SetDefaultNXT(open); % line 4
% So, our actual code begins here:
% We want our robot to move straight forward with 20% of its maximum
% speed:
mAC = NXTMotor ('AC', 'Power', 20);
% We open the ultrasonic sensor:
OpenUltrasonic (SENSOR_1);
% Once we have set the speed and opened the ultrasonic sensor, we
% make the robot move at the desired speed:
mAC.SendToNXT ();
% Open ultrasonic sensor:
OpenUltrasonic (SENSOR_1);
% Opens left light sensor:
OpenLight (SENSOR_2,'ACTIVE');
% Opens right light sensor
OpenLight (SENSOR_3,'ACTIVE');
% Both light sensors will detect if the robot is encountering the
black line. And the ultrasonic sensor will detect if the robot
encounters an object.
% We detect the surface and we store the obtained value which
represents the colour (in our case: either black or white). We want to
store these values and we store these values in a vector. So, we also
need variable which will the index of our vector. Here we initialize
the variable which will be the index of our vector:
vec_index = 1;
minDistance = 25;
maxDistance = 100;
averageDistance = 50;
```

```

x = 1
while (x > 0)
    % Reads sensor and puts the value into a vector
    val(vec_index) = GetUltrasonic (SENSOR_1);
    % If the distance is greater than the max distance
    % increase speed so that the robot is not far behind
    % the object.
    mAC = NXTMotor ('AC', 'Power', 15);
    mAC.SendToNXT(); % Motors make the robot move
    % We want to obtain values of the surface to detect if the
    % robot is encountering the black line. We get the values from
    % both sensors. We store both values in two different vectors.
    % One vector is for the left light sensor:
    vec_val_2(vec_index) = GetLight(SENSOR_2); % Left sensor
    % And the other vector is for the right light sensor:
    vec_val_3(vec_index) = GetLight(SENSOR_3); % Right sensor

    % If the left light sensors detects black:
    if vec_val_2(vec_index) < 400
        % The robot will first move backwards. We achieve this by
        % using a negative value for the speed. The robot will
        % move backwards with a speed of this value multiplied by
        % minus one (-20 * -1 = 20):
        mAC = NXTMotor('AC', 'Power', -20);
        % Now, the robot moves backwards with a speed of 20:
        mAC.SendToNXT ();
        pause(1);
        % Robot turns
        mA = NXTMotor('A', 'Power', 15);
        mC = NXTMotor('C', 'Power', 30);
        mA.SendToNXT ();
        mC.SendToNXT ();
        pause (1);
    end % End if statement

    % If the value represents black
    if vec_val_3 (vec_index) < 400
        mAC = NXTMotor ('AC', 'Power', -20);
        mAC.SendToNXT ();
        pause(1);
        mC = NXTMotor ('C', 'Power', 15);
        mA = NXTMotor ('A', 'Power', 30);
        mC.SendToNXT ();
        mA.SendToNXT ();
        pause (1)
    end % End if statement

```

```

if val(vec_index) > maxDistance
    mAC = NXTMotor ('AC', 'Power', 40);
    mAC.SendToNXT ();
end % End if statement

% If the distance is less than the minimum distance
% first check whether the object has stopped or slowed down
if val(vec_index) <= minDistance
    mAC.Stop ('brake');
    pause(3);
    pause_distance = GetUltrasonic (SENSOR_1);
    % If the distance is the same or less than the new
    % distance read the object has stopped. Therefore the robot
    % must stop as well.
    if pause_distance <= minDistance
        mAC.Stop ('off')
        x = 0;
    end % End if statement
end % End if statement
vec_index = vec_index + 1;
pause(0.1);
end %End while loop

```

Exercise 3: Follow a line Write a program in pseudocode and in MATLAB that commands the robot to follow a curved black line, and then stops at a T-junction. Explain how your algorithm works and evaluate it. Is it smooth, is it fast, is it slow, just good?

Answer: This is our pseudocode:

Calibrate robot:

 Determine what value is considered as the colour black

 Determine what value is considered as the colour white

Position the robot on the black line

While (true)

 While (Both sensors detect white)

 Drive straight forward

 End_while

 If (Left sensor sees black)

 Then turn left

 If (Right sensor sees black)

 Then turn right

 If (Both sensors see black)

 Then stop moving

End_while

Algorithm explanation:

Function setup

End function

function CALIBRATE

 Place robot's left sensor on white

 Open light left sensor

 Value of white = Read Light Sensor

 Obtain a lower bound limit for the value of white

 That value is now considered as being the value for white

 Place robot's left sensor on black

 Open light left sensor

 Value of black = Read Light Sensor

 Obtain an upper bound limit for the value of black

 That value is now considered as being the value for black

done function

Program:

We use variable `vec_index` as the index of our vector where we store the obtained values from both light sensors

`vec_index` is an integer and we increment `vec_index` each time the while loop gets executed

While loop:

Repeat forever (while not encountering T-junction):

Step 1

Read value from left sensor and place it in a vector as an element with index

`vec_index`

Read value from right sensor and place it a vector as an element with index

`vec_index`

End step 1

Step 2

Start both motors (in the code: `mAC`) the same constant speed. We do this by using:

```
mAC = NXTMotor (AC, Power, 15);
```

```
mAC.SendToNXT;
```

Step 3

If the value from the variable of the left sensor is less than the value of black

Turn right (Reduce speed of right motor by half or more)

Return back to step 1

If the value from the variable of the right sensor is less than the value of black

Turn left (Reduce speed of left motor by half or more)

Return back to step 1

If the values from both variables of the sensors is greater than the value of white

Remain straight(motors go at the same speed)

Return back to step 1

If the values from the both variables of the sensors is less than the value of black

Stop both motors

End repeat

Evaluation of algorithm:

Our algorithm is slow as, although our algorithm achieves its result, by following the line on a curvy path and stopping at the T-Junction. It does not drive quickly nor does it calculate the error obtained, which is how far the robot has deviated off the track, and adjusts the different motor speeds using the PID (Proportion, Integral and Derivative). A smoother and faster robot would have been obtained using this method.

Answer : This is our MATLAB code

```
% We used the variables val_white and val_black which we determined  
% by calibration (as we described earlier):
```

```
% Instead of the following four lines indicated as  
% line 1, 2, 3, and 4, we use the command '>> setup Exercise 3'.  
% COM_CloseNXT('all');           % line 1  
% close all;                     % line 2  
% open = COM_OpenNXT();          % line 3
```

```

% COM_SetDefaultNXT(open); % line 4
% So, our actual code begins here:
% Opens left light sensor:
OpenLight (SENSOR_2,'ACTIVE');
% Opens right light sensor
OpenLight (SENSOR_3,'ACTIVE');
% Both sensors detect if the robot is encountering the black line.
% We detect the surface and we store the obtained value which
% represents the colour (in our case: either black or white). We want
% to store these values and we store these values in a vector. So, we
% also need variable which will the index of our vector. Here we
% initialize the variable which will be the index of our vector:
vec_index = 1;
while (true)
    % We want our robot to move straight forward with 15% of its
    % maximum speed:

    mAC = NXTMotor ('AC', 'Power', 15);
    % The motors make the robot move:
    mAC.SendToNXT();
    % We want to obtain values of the surface to detect if the
    % robot is encountering the black line. We get the values from
    % both sensors. We store both values in two different vectors.
    % One vector is for the left light sensor:
    vec_val_2(vec_index) = GetLight(SENSOR_2); % Left sensor
    pause (0.1);
    % And the other vector is for the right light sensor:
    vec_val_3(vec_index) = GetLight(SENSOR_3); % Right sensor

    % If the left light sensors detects black:
    if vec_val_2(vec_index) < 400
        % Robot turns right
        mA = NXTMotor('A', 'Power', 5);
        mC = NXTMotor('C', 'Power', 15);
        mA.SendToNXT ();
        mC.SendToNXT ();
        pause (1);
    end % End if statement

    % If the right light sensors detects black:
    if vec_val_3 (vec_index) < 400
        % Robot turn left
        mC = NXTMotor ('C', 'Power', 5);
        mA = NXTMotor ('A', 'Power', 15);
        mC.SendToNXT ();
        mA.SendToNXT ();
        pause (1);
    end
end

```

```

end % End if statement

% If both light sensors detects white:
if vec_val_2(vec_index) < 600
    if vec_val_3(vec_index)<600
        % Robot moves straight forward
        mAC = NXTMotor('AC', 'Power', 15);
        mAC.SendToNXT ();
        pause (1);
    end % End if statement
end % End if statement

% If both light sensors detects black:
if vec_val_2(vec_index) < 400
    if vec_val_3(vec_index)< 400
        % Robot stops
        mAC.Stop ('off');
        pause (1);
    end % End if statement
end % End if statement

vec_index = vec_index + 1;
pause (0.1);

end % End while loop

```