**Faculty of Engineering – University of Ottawa**

**SEG4145 REAL-TIME AND EMBEDDED SOFTWARE DESIGN**

**Yuanyuan Li – 7499879 – yli289@uottawa.ca**

**Vincent Maliko – 4225528 – vmali033@uottawa.ca**

**Rémi Lacroix – 7528357 – rlacr062@uottawa.ca**

**January 30, 2014**

Report for lab 1:

Introduction to the Arduino IDE

And the Robotic Platform

**Date of experiment:** Thursday January 16, 2014

**Date due:** Thursday January 30, 2014

Table of Contents

[1. Objectives 2](#_Toc378845505)

[2. Tasks, issues and discussion 2](#_Toc378845506)

[2.1 Calibration 2](#_Toc378845507)

[2.1.1 Making the robot turn 2](#_Toc378845511)

[2.1.2 Making the robot travel a straight line 3](#_Toc378845512)

[2.1.3 Calibration method 3](#_Toc378845513)

[2.2 Requirements 4](#_Toc378845514)

[2.2.1 LCD display 5](#_Toc378845515)

[2.2.2 LED 5](#_Toc378845516)

[2.2.3 Bloc motor-wheel 5](#_Toc378845517)

[3. High Level Design 6](#_Toc378845518)

[3.1 Modular 6](#_Toc378845519)

[3.2 Structure and algorithm 8](#_Toc378845520)

[4. Conclusion 9](#_Toc378845521)

[Annexe – source code 10](#_Toc378845522)

[Annexe I – main source code 10](#_Toc378845523)

[Annexe II – test calibration of the robot 16](#_Toc378845524)

# Objectives

* Implement and download a software program for the robot using the Arduino IDE.
* Manipulate the LCD display, LED light and wheels of the robot.
* Manipulate the LCD display of the robot.
* Manipulate the LED light of the robot.
* Manipulate the wheels of the robot to move the robot forwards, backwards and rotate the robot clockwise and counterclockwise.

# Tasks, issues and discussion

## Calibration

First write a program that continuously makes the robot rotate in one direction (either clockwise or counter clockwise). Time how long it takes for the robot to perform a full rotation and use that value to approximate angular rotations. For example, if it takes 10s for a rotation, then a 360° takes 10s, 180° takes 5s, 90° takes 2.5s, etc. This step is being performed since all motors are different so the rotation times for each robot might be slightly different.



### Making the robot turn

**Step 1: Computing the delay required for a robot to make a full rotation (360o)**

In order to calibrate the robot’s movements, we wrote a program to establish some values. We made the robot rotate clockwise and counter clockwise. We then recorded several values for a full rotation. We calculated an average based on the sample values previously recorded and launch the program using that average, which was adjusted later on to a few more milliseconds. The team concluded an average value of approximately 4 seconds was required to complete a full turn.

**Step 2: Computing the delay for 90o and45o**

Once we were able to time how long it took for the robot to perform a full rotation. That value was divided by 4 and 8 in order to get the time for 90o and45o. We tested each unit of time and made some adjustments accordingly.

On January 23rd in a free session, we used the robot and it appeared that the robot was no longer making a full turn in 4 seconds and required more time (4.4 seconds). A fortioridelay for 90° and 45° were no longer valid either.

We will give more explanation about what happened in the discussion. But after these results we concluded we need to calibrate the robot each new lab session.

### Making the robot travel a straight line

**Step 1: Calibration of the left and right wheels**

Since each motor was travelling at its own speed, some calibration was required in order to have both wheels move at the same speed.

The team tried several values, and was able to narrow those values down to 191 (left motor), and 10 (right motor). As a result both wheels travelled at equal speed which allowed the robot to travel in a straight line. Even though the overall movement was good enough, the wheels were shimmying at times.

However, we faced the same issue encountered for the full rotation in the free session. On January 23rd, the values were no longer allowing for the robot to travel a straight line. Some calibration was required. After trying a few values, we came to a conclusion that 184.9 (left motor) and 14 (right motor).

?? Like for the rotation, we need to calibrate the pulse width value of each wheels for next lab session.

**Step 2: Differentiation between moving forward and backward**

We observed that values determined for travelling forward were not good for travelling backward. We thought by switching both left and right wheels' values would have been enough to revert the robot's movement in straight line. This turned out to be wrong. Values were not exactly the same. So we needed to find specific values for the two way and for each motor: 4 motor’s value. These values are good for a session, we need to check and modify them every time.

### Calibration method

After the first calibration, for next time we start with these value and we adjust them with a binary research. The process is long but we write final calibration value for each session: database.

Figure 1: Table of calibration value

The program written for this part is available in **annexe II**. Structure and details about it are given in the high level design.

## Requirements

Write a program for the robot that performs the following tasks:

• Display the two student numbers of the students for 5 seconds and have the robot traverse the following paths described below:

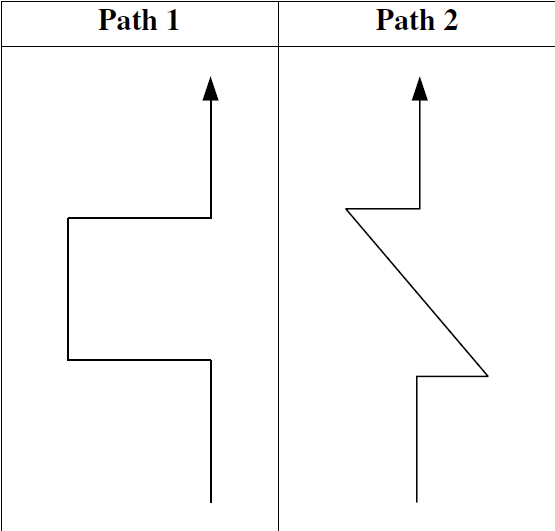


Figure 2: Path to be performed by the robot

**Note:** Each line segment should take 10 seconds to travel. The rotations in paths 1 are 90°. All rotations in paths 2, are 45° (or the appropriate multiple of 45°) from the horizontal.

The following characteristics must be exhibited on the robot as the program is operating:

* Display one student number on each line of the LCD display at the beginning of the program for 5 seconds.
* As the student numbers are being displayed, the LED light must flash/flicker after each second to indicate the length of time that has passed
* The message “Path 1” and “Path 2” must be displayed on the top line of the LCD screen for 3 seconds before the respective path is traversed. The bottom line of the LCD screen must be blank.
* All text displayed on the LCD display must be centred on the top and bottom lines.
* The following messages must be displayed on the LCD display as the robot makes its various movements:
  + moving forward
  + moving backward
  + rotating left
  + rotating right
  + stopped

Each message with two words must display the first word on the top line and the second word on the bottom line. A message with one word must be displayed entirely on the top line with a blank bottom line.

Your program should be as modular as possible. The following steps are **highly** recommended in the development of this lab:

* Defining macros in the header file for both motors and all possible directions (i.e. one macro for when the left motor is stopped, one macro for when the right motor rotates backwards, etc.)
* Defining functions to stop the robot, rotate it clockwise or counter clockwise, and moving it forward and backwards.

**BONUS:** After traversing a path, the robot must pause for approximately 5 seconds, and traverse the same path **backwards**, ending up in approximately the same position it started. 2.5% will be added for every path that can be correct traversed backwards for a maximum bonus of 5% overall.

From requirements, we could define different steps and modules to structure our program. We divided our work according to the type of components:

* LCD display
* LED (light-emitting diode)
* Bloc motor-wheel

Specific functions were designed for these components. Each components were used to implement the paths described above.

### LCD display

*We need to print text on the display. The LCD screen is composed of two lines of 16 characters. The first line is called the top line and the other one the bottom line. All text displayed must be centred on the line. And it should be possible to parameterise the display time.*

### LED

The LED light must flash/flicker after each second to indicate the length of time that has passed.

### Bloc motor-wheel

A motor have three states: it does not turn, it rotates back or forth. So we can define a macro, a basic function for each state of the left and the right motor.

Then to perform paths, the robot must have several movements. It should move forward and backward, rotate to the left or to the right and keep still. These movements will be implement in the form of functions.

We were able to print on the display however two straight lines kept appearing on each side of the first row.

# High Level Design

## 3.1 Modular

## 3.1.1 LCD Functions

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* LCD functions

\*

\* lcdDisplay(String topLine, String bottomLine)

\* lcdPosition(int row, int col)

\* lcdClear()

\* backlightOn()

\*/

/\* Name : lcdPosition function which allow to place the cursor on a specific position (row, col)

\* Param In : row , int variable , row=0 : top line / row=1 : bottom line

\* col , int variable , position on the line / range 0-15

\* Param Out :

\* Return :

\*/

void lcdPosition**(**int row**,** int col**){**

LCD**.**write**(**0xFE**);** // put LCD in command mode

LCD**.**write**((**col **+** row**\***64 **+** 128**));** // place the cursor in the desired position

delay**(**10**);**

**}**

/\* Name : lcdClear function which allow to clear the LCD screen

\* Param In :

\* Param Out :

\* Return :

\*/

void lcdClear**(){**

LCD**.**write**(**0xFE**);** // put LCD in command mode

LCD**.**write**(**0x01**);** // clear the LCD screen

delay**(**10**);**

**}**

/\* Name : backlightOn function which allow to turn on the back light

\* Param In :

\* Param Out :

\* Return :

\*/

void backlightOn**(){**

LCD**.**write**(**0x7C**);** // command flag for backlight stuff

LCD**.**write**(**0x9D**);** // turn on back light

delay**(**10**);**

**}**

/\* Name : lcdDisplay function which allow to display text on the LCD screen

\* Param In : topLine , String variable , text displayed on the top line of the LCD screen

\* bottomLine , string variable , text displayed on the bottom line of the LCD screen

\* Param Out :

\* Return :

\*/

void lcdDisplay**(**String topLine**,** String bottomLine**){**

// Calculation of the line offset for both line of the LCD screen

int topLineOffset **=** int**((**16**-**topLine**.**length**())/**2**);**

int bottomLineOffset **=** int**((**16**-**topLine**.**length**())/**2**);**

// Clear the LCD screen

backlightOn**();**

//displayOn();

lcdClear**();**

// Print the top line text

**if(**topLine**.**length**()** **>** 0**){**

lcdPosition**(**0**,** topLineOffset**);**

LCD**.**print**(**topLine**);** // or try with write() !!!

**}**

// Print the bottom line text

**if(**bottomLine**.**length**()** **>** 0**){**

lcdPosition**(**1**,** bottomLineOffset**);**

LCD**.**print**(**bottomLine**);**

**}**

**}**

**3.1.2 LED Functions**

**(copy paste the code)**

**3.1.2 Move Functions**

**Moving forward (copy paste the code)**

**Moving backward**

**Turn left**

**Turn right**

**Stop**

Explain every function and macro and the structure of the program

**LED**

For the moving, because we use several call of same basic fonctions, we define macro which will be call in high level function:

Macros simple action;

Functions high function, add lcd function

**Moving forward**

**Moving backward**

**Turn left**

**Turn right**

**Stop**

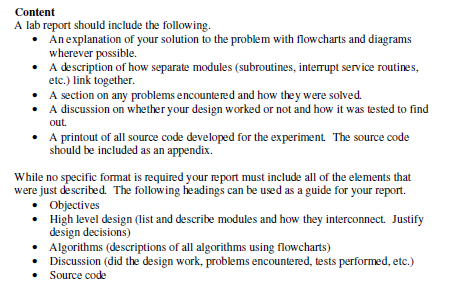
## 3.2 Structure and algorithm

We design our main program as following. We have a call of the header file, a bloc for variables and parameters. Th are parameters for the control of the LCD, the LED and

Figure 3: Structure of the main program

# 4. Conclusion

It is impossible to travel a perfect straight line by using manual calibration for each wheel. A better solution would be to calculate the pace of each wheel and readjust on the fly as the movement occurs. Which means constantly probing and checking whether the speed of both is equal, if not lower the one that moves faster or increase the one that moves slower.



# Annexe – source code

## Annexe I – main source code

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Names: Vincent Maliko

\* Yuanyuan Li 7499879

\* Rémi Lacroix 7528357

\*

\* Course Code: SEG 4145

\* Lab Number: 1

\* File name: lab1\_script

\* Date: January 16, 2014

\*

\*

\* Description

\* \*\*\*\*\*\*\*\*\*\*\*\*\*

\* This current file is the file template of the lab #1:

\* Introduction to the Arduino IDE and the Robotic Platform

\*

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

// Header files

#include <SoftwareSerial.h>

// Global variables (fixed)

int timer **=** 1000**;** // definition of a time unit 1000s

int factor **=** 8**;** // defintion of a multiplicator factor

int motor\_left\_pin **=** 2**;**

int motor\_right\_pin **=** 4**;**

int pulse\_left\_rotation **=** 10**;** // do not change these values

int pulse\_right\_rotation **=** 191.5**;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Calibration values (change them according experimental results)

// Rotating

int oneRound **=** 4500**;** //3780; // time en ms for a robot's rotation

// Straight line

int pulse\_left\_forward **=** 185**;**

int pulse\_right\_forward **=** 10**;**

int pulse\_left\_backward **=** 5**;**

int pulse\_right\_backward **=** 200**;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// global variables (suite)

int board\_LED **=** 13**;**

int LCD\_display **=** 18**;**

SoftwareSerial LCD **=** SoftwareSerial**(**0**,** LCD\_display**);**

String topLine**=** ""**;**

String bottomLine**=** ""**;**

int rotation\_45 **=** oneRound**/**8**;** // unit of rotation

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Macros of all possible directions of both motors

\*

\* Directions : stop, forward, backward

\* Motors : left, right

\*/

#define MacroStopMotorLeft() analogWrite(motor\_left\_pin, 0)

#define MacroStopMotorRight() analogWrite(motor\_right\_pin, 0)

#define MacroForwardMotorLeft() analogWrite(motor\_left\_pin, pulse\_left\_forward)

#define MacroForwardMotorRight() analogWrite(motor\_right\_pin, pulse\_right\_forward)

#define MacroBackwardMotorLeft() analogWrite(motor\_left\_pin, pulse\_left\_backward)

#define MacroBackwardMotorRight() analogWrite(motor\_right\_pin, pulse\_right\_backward)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// The setup routine runs once when you press reset:

// This function is called before the loop() function

void setup**(){**

// Initialization instructions

// Sets the digital pin as output

pinMode**(**motor\_left\_pin**,** OUTPUT**);**

pinMode**(**motor\_right\_pin**,** OUTPUT**);**

pinMode**(**board\_LED**,**OUTPUT**);**

pinMode**(**LCD\_display**,** OUTPUT**);**

// Open the serial port at 9600 bps

Serial**.**begin**(**9600**);**

// Open the serial port to write data at 9600 bps

LCD**.**begin**(**9600**);**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// The loop routine runs over and over again forever:

void loop**(){**

// The instructions executed to perform the main

// functionality of the program

// display student number

lcdDisplay**(**"7499879"**,** "7528357"**);**

//Loop for the led - total time 5000s

**for** **(**int i **=** 0**;** i **<** 5**;** i**++)** **{**

digitalWrite**(**board\_LED**,** HIGH**);** // turn the LED on

delay**(**timer**);**

digitalWrite**(**board\_LED**,** LOW**);** // turn the LED off

delay**(**timer**);**

**}**

// Clear the lCD screen

lcdClear**();**

// Wait few second (5s)

delay**(**5000**);**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// path 1

lcdDisplay**(**"path 1"**,** ""**);**

delay**(**3000**);**

// instructions

moveForward**(**factor**\***timer**);**

rotateClockwise**(**2**\***rotation\_45**);** // turn left

moveForward**(**factor**\***timer**);**

rotateCounterClockwise**(**2**\***rotation\_45**);** // turn right

moveForward**(**factor**\***timer**);**

rotateCounterClockwise**(**2**\***rotation\_45**);** // turn right

moveForward**(**factor**\***timer**);**

rotateClockwise**(**2**\***rotation\_45**);** // turn left

moveForward**(**factor**\***timer**);**

stopped**();**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// path 1 reverse

/\*

lcdDisplay("path 1", "reverse");

delay(3000);

// instructions

moveBackward(factor\*timer);

rotateCounterClockwise(2\*rotation\_45); // turn right

moveBackward(factor\*timer);

rotateClockwise(2\*rotation\_45); // turn left

moveBackward(factor\*timer);

rotateClockwise(2\*rotation\_45); // turn left

moveBackward(factor\*timer);

rotateCounterClockwise(2\*rotation\_45); // turn right

moveBackward(factor\*timer);

stopped();

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// path 2

lcdDisplay**(**"path 2"**,** ""**);**

delay**(**3000**);**

// instructions

moveForward**(**factor**\***timer**);**

rotateCounterClockwise**(**2**\***rotation\_45**);** // turn right (90°)

moveForward**(**factor**\***timer**/**2**);**

rotateClockwise**(**3**\***rotation\_45**);** // turn left (135°)

moveForward**(**factor**\***timer**);**

rotateCounterClockwise**(**3**\***rotation\_45**);** // turn right (135°)

moveForward**(**factor**\***timer**);**

rotateClockwise**(**2**\***rotation\_45**);** // tunr left (90°)

moveForward**(**factor**\***timer**);**

stopped**();**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// path 2 reverse

/\*

lcdDisplay("path 2", "reverse");

delay(3000);

// instructions

moveBackward(factor\*timer);

rotateCounterClockwise(2\*rotation\_45); // turn right (90°)

moveBackward(factor\*timer);

rotateClockwise(3\*rotation\_45); // turn left (135°)

moveBackward(factor\*timer);

rotateCounterClockwise(7\*rotation\_45); // turn right (315°)

moveBackward(factor\*timer);

rotateClockwise(2\*rotation\_45); // turn left (90°)

moveBackward(factor\*timer);

stopped();

\*/

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Moving functions

\*

\* stopped(int t\_delay)

\* moveForward(int t\_delay)

\* moveBackward(int t\_delay)

\* rotateClockwise(int t\_delay)

\* rotateCounterClockwise(int t\_delay)

\*/

/\* Name : stopped function which stops the robot all the time

\* Param In : t\_delay, int variable, time parameter

\* Param Out :

\* Return :

\*/

void stopped**(){**

lcdDisplay**(**"stopped"**,** ""**);**

MacroStopMotorLeft**();**

MacroStopMotorRight**();**

**}**

/\* Name : moveForward function which allow to move the robot forward for a delay given in parameter

\* Param In : t\_delay, int variable, time parameter

\* Param Out :

\* Return :

\*/

void moveForward**(**int t\_delay**){**

lcdDisplay**(**"moving"**,** "forward"**);**

MacroForwardMotorLeft**();**

MacroForwardMotorRight**();**

delay**(**t\_delay**);**

**}**

/\* Name : moveBackward function which allow to move the robot backward for a delay given in parameter

\* Param In : t\_delay, int variable, time parameter

\* Param Out :.

\* Return :

\*/

void moveBackward**(**int t\_delay**){**

lcdDisplay**(**"moving"**,** "backward"**);**

MacroBackwardMotorLeft**();**

MacroBackwardMotorRight**();**

delay**(**t\_delay**);**

**}**

/\* Name : rotateClockwise function which allow to rotate the robot clockwise for a delay given in parameter

\* Param In : t\_delay, int variable, time parameter

\* Param Out :

\* Return :

\*/

void rotateClockwise**(**int t\_delay**){** // rotate to the right

lcdDisplay**(**"rotating"**,** "right"**);**

analogWrite**(**motor\_left\_pin**,** pulse\_right\_rotation**);**

analogWrite**(**motor\_right\_pin**,** pulse\_right\_rotation**);**

delay**(**t\_delay**);**

**}**

/\* Name : rotateCounterClockwise function which allow to rotate the robot counter clockwise for a delay given in parameter

\* Param In : t\_delay, int variable, time parameter

\* Param Out :

\* Return :

\*/

void rotateCounterClockwise**(**int t\_delay**){** // rotate to the left

lcdDisplay**(**"rotating"**,** "left"**);**

analogWrite**(**motor\_left\_pin**,** pulse\_left\_rotation**);**

analogWrite**(**motor\_right\_pin**,** pulse\_left\_rotation**);**

delay**(**t\_delay**);**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* LCD functions

\*

\* lcdDisplay(String topLine, String bottomLine)

\* lcdPosition(int row, int col)

\* lcdClear()

\* backlightOn()

\*/

/\* Name : lcdPosition function which allow to place the cursor on a specific position (row, col)

\* Param In : row , int variable , row=0 : top line / row=1 : bottom line

\* col , int variable , position on the line / range 0-15

\* Param Out :

\* Return :

\*/

void lcdPosition**(**int row**,** int col**){**

LCD**.**write**(**0xFE**);** // put LCD in command mode

LCD**.**write**((**col **+** row**\***64 **+** 128**));** // place the cursor in the desired position

delay**(**10**);**

**}**

/\* Name : lcdClear function which allow to clear the LCD screen

\* Param In :

\* Param Out :

\* Return :

\*/

void lcdClear**(){**

LCD**.**write**(**0xFE**);** // put LCD in command mode

LCD**.**write**(**0x01**);** // clear the LCD screen

delay**(**10**);**

**}**

/\* Name : backlightOn function which allow to turn on the back light

\* Param In :

\* Param Out :

\* Return :

\*/

void backlightOn**(){**

LCD**.**write**(**0x7C**);** // command flag for backlight stuff

LCD**.**write**(**0x9D**);** // turn on back light

delay**(**10**);**

**}**

/\* Name : lcdDisplay function which allow to display text on the LCD screen

\* Param In : topLine , String variable , text displayed on the top line of the LCD screen

\* bottomLine , string variable , text displayed on the bottom line of the LCD screen

\* Param Out :

\* Return :

\*/

void lcdDisplay**(**String topLine**,** String bottomLine**){**

// Calculation of the line offset for both line of the LCD screen

int topLineOffset **=** int**((**16**-**topLine**.**length**())/**2**);**

int bottomLineOffset **=** int**((**16**-**topLine**.**length**())/**2**);**

// Clear the LCD screen

backlightOn**();**

//displayOn();

lcdClear**();**

// Print the top line text

**if(**topLine**.**length**()** **>** 0**){**

lcdPosition**(**0**,** topLineOffset**);**

LCD**.**print**(**topLine**);** // or try with write() !!!

**}**

// Print the bottom line text

**if(**bottomLine**.**length**()** **>** 0**){**

lcdPosition**(**1**,** bottomLineOffset**);**

LCD**.**print**(**bottomLine**);**

**}**

**}**

# Annexe II – test calibration of the robot

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Names: Vincent Maliko 4225528

\* Yuanyuan Li 7499879

\* Rémi Lacroix 7528357

\*

\*

\* Course Code: SEG 4145

\* Lab Number: 1

\* File name: lab1\_calibration

\* Date: January 16, 2014

\*

\*

\* Description

\* \*\*\*\*\*\*\*\*\*\*\*\*\*

\* Introduction to the Arduino IDE and the Robotic Platform

\*

\* This program allows to calibrate robot's movement : rotation, moving forward and backward

\* A way to calibrate the robot is to experiment different calibration's values

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

// Header files

#include <SoftwareSerial.h>

// Global variables

int timer **=** 1000**;** // definition of a time unit 1000s

int factor **=** 8**;** // defintion of a multiplicator factor

int motor\_left\_pin **=** 2**;**

int motor\_right\_pin **=** 4**;**

int pulse\_left\_rotation **=** 10**;** // do not change these values

int pulse\_right\_rotation **=** 191.5**;**

// Calibration values (change them according experimental results)

// Rotating

int oneRound **=** 4500**;** //3780; // time en ms for a robot's rotation

// Straight line

int pulse\_left\_forward **=** 185**;**

int pulse\_right\_forward **=** 10**;**

int pulse\_left\_backward **=** 5**;**

int pulse\_right\_backward **=** 200**;**

// The setup routine runs once when you press reset:

// This function is called before the loop() function

void setup**(){**

// Initialization instructions

// Sets the digital pin as output

pinMode**(**motor\_left\_pin**,** OUTPUT**);**

pinMode**(**motor\_right\_pin**,** OUTPUT**);**

// Open the serial port at 9600 bps

Serial**.**begin**(**9600**);**

**}**

// The loop routine runs over and over again forever:

void loop**(){**

// The instructions executed to perform the main

// functionality of the program

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Test calibration rotation

delay**(**timer**);** // waiting 5s before to the next instruction

rotateCounterClockwise**(**oneRound**);** // rotation to the left

stopped**(**5**\***timer**);**

rotateClockwise**(**oneRound**);** // rotation to the right

stopped**(**timer**);**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Test calibration moving forward and backward

delay**(**timer**);** // waiting 5s before to the next instruction

moveStraightLine**(**8**\***timer**,** 185**,** 10**);** // moving forward

stopped**(**timer**);**

moveStraightLine**(**8**\***timer**,** 5**,** 200**);** // moving backward

stopped**(**timer**);**

// 5 time loop

**for** **(**int i **=** 0**;** i **<** 5**;** i**++)** **{**

// insert the instructions

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Functions

\*

\* stopped(int t\_delay)

\* rotateClockwise(int t\_delay)

\* rotateCounterClockwise(int t\_delay)

\* moveStraightLine(double val\_left, double val\_right)

\*/

/\* Name : stopped function which stops the robot for a delay given in parameter

\* Param In : t\_delay, int variable, time parameter

\* Param Out :

\* Return :

\*/

void stopped**(**int t\_delay**){**

analogWrite**(**motor\_left\_pin**,** 0**);**

analogWrite**(**motor\_right\_pin**,** 0**);**

delay**(**t\_delay**);** // waiting 't\_delay' ms before to the next instruction

**}**

/\* Name : rotateClockwise function which allow to rotate the robot clockwise for a delay given in parameter

\* Param In : t\_delay, int variable, time parameter

\* Param Out :

\* Return :

\*/

void rotateClockwise**(**int t\_delay**){** // rotate to the right

analogWrite**(**motor\_left\_pin**,** pulse\_right\_rotation**);**

analogWrite**(**motor\_right\_pin**,** pulse\_right\_rotation**);**

delay**(**t\_delay**);** // waiting 't\_delay' ms before to the next instruction

**}**

/\* Name : rotateCounterClockwise function which allow to rotate the robot counter clockwise for a delay given in parameter

\* Param In : t\_delay, int variable, time parameter

\* Param Out :

\* Return :

\*/

void rotateCounterClockwise**(**int t\_delay**){** // rotate to the left

analogWrite**(**motor\_left\_pin**,** pulse\_left\_rotation**);**

analogWrite**(**motor\_right\_pin**,** pulse\_left\_rotation**);**

delay**(**t\_delay**);** // waiting 't\_delay' ms before to the next instruction

**}**

/\* Name : moveStraightLine function which allow to move the robot - in a way defined by two parameter - for a delay given in parameter

\* Param In : t\_delay , int variable , time parameter

\* val\_left , double variable , pulse width value for the left motor

\* val\_right , double variable , pulse width value for the right motor

\* Param Out :

\* Return :

\*/

void moveStraightLine**(**int t\_delay**,** double val\_left**,** double val\_right**){**

analogWrite**(**motor\_left\_pin**,** val\_left**);**

analogWrite**(**motor\_right\_pin**,** val\_right**);**

delay**(**t\_delay**);** // waiting 't\_delay' ms before to the next instruction

**}**