**Université d’Ottawa**
Faculté de Génie,
École de science informatique
et de génie électrique

uOttawa
L’Université canadienne
Canada’s university

**University of Ottawa**
Faculty of Engineering
School of Electrical Engineering and
Computer Science

# REAL TIME AND EMBEDDED SOFTWARE DESIGN

SEG 4145

LABORATORY 1
Introduction to the Arduino IDE and the Robotic Platform

Winter 2014

# Lab 1: Introduction to the Arduino IDE and the Robotic Platform

**Objectives:**

On completion of this lab, students will be able to perform the following tasks:

- Implement and download a software program for the robot using the Arduino IDE.
- 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.

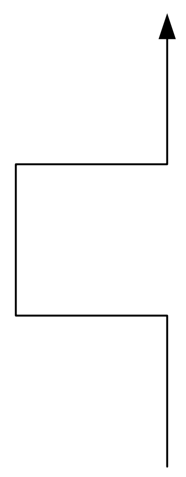
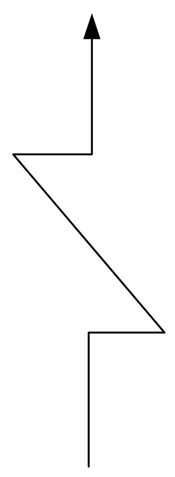
**Tasks:**

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 full 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.

HINT: Your robot can be made to 'rotate' by moving the wheels in opposite directions at the same time.

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:

| **Path 1** | **Path 2** |
|---|---|
|  |  |

**Note:** Each line segment should take 5 seconds to travel. The rotations in path 1 are 90°. All rotations in path 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 centered 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 stop 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 counterclockwise, 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.

## *Main Structure of a Program*

A program written for the Arduino program takes the following general form:

```
// the setup routine runs once when you press reset:
// This function is called before the loop() function
void setup() {
   // Initialization instructions
}

// the loop routine runs over and over again forever:
void loop() {
  // The instructions executed to perform the main
  // functionality of the program
}
```

Each program must contain the following functions:
**setup():** This function is used to initialize the Arduino program through a sequence of instructions that are executed one time.
**loopO:** This function is used to implement the main functionality of the program. This function is called endlessly to perform the execution of a program.

Other functions may be defined and called from the setup() and loop() functions as required.

## *Arduino Board LED Manipulation*

The Arduino board LED is connected to digital pin # 13 (see Appendix A of the manual). The LED Pin should be set as OUTPUT and you can only write to this pin. The LED pin values may be HIGH or LOW.

The program demonstrated below illustrates the steps required for the initialization and use of LED. It turns on the LED for one second, then off for one second, repeatedly.

```
// LED constant
#define BOARD_LED 13

void setup() {
  // Initialize LED pin as an output pin.
  pinMode(BOARD_LED, OUTPUT);

}

void loop() {
  // turn the LED on (HIGH is the voltage level)
  digitalWrite(BOARD_LED, HIGH);
  // wait for a second
  delay(1000);
  // turn the LED off by making the voltage LOW
  digitalWrite(BOARD_LED, LOW);
  // wait for a second
  delay(1000);
}
```

## *Motor Manipulation*

Pulse width modulation is a technique used to control the power sent to devices like the motors on the robot. A voltage wave with specific characteristics must be transmitted to each motor in order for the motor to move in a specific direction. Here is an initial code to make the left wheel rotate backwards:

```
// Left Motor constant
#define LEFT_MOTOR 2

// Direction constants
#define LEFT_BACKWARD 10

void setup() {
  // Initialize all pins.
  pinMode(LEFT_MOTOR, OUTPUT);
}

void loop() {
  // move left wheel backward
  backward();
}

void backward() {
  analogWrite (LEFT_MOTOR, LEFT_BACKWARD);
}
```

Consult Appendix A for motor pin values and the appropriate pulse width values. **Please note that the motor constants given in appendix A the typical values found on a sample robot. You may need to change these values to make the wheels rotate approximately with the same speed.**

## *LCD Screen Manipulation*

A variable of the type ‘SoftwareSerial’ must be created to manipulate the LCD screen. The pin for the LCD screen must be specified with the initialization of this variable. The serial port must also be initialized in a similar manner to a serial port where a data transmission speed must be specified as its operation is started. An example of the variable creation and initialization is shown below:

```
#include <SoftwareSerial.h>

#define lcd_pin_number = 18;
SoftwareSerial LCD(0, lcd_pin_number);
LCD.begin(9600); // Open the serial port to
                 // write data at 9600 bps
```

Data is displayed on the LCD screen with the **print** function which is used in the following manner:

```
 LCD.print("Hello world!");
```

Commands are single integer values written with the **write** function. The LCD screen must be put into a command execution mode before the desired instruction can be executed. It is also highly recommended that a small delay is executed after in case numerous instructions are executed to limit the amount of data transmitted to the LCD screen at one time. An example of the instructions required to clear the LCD screen are demonstrated below:

```
LCD.write(0x7C);  // Put the LCD screen in command mode.
LCD.write(0x01);  // Clear the LCD screen.
delay(10);        // Delay operation for 10 ms
```

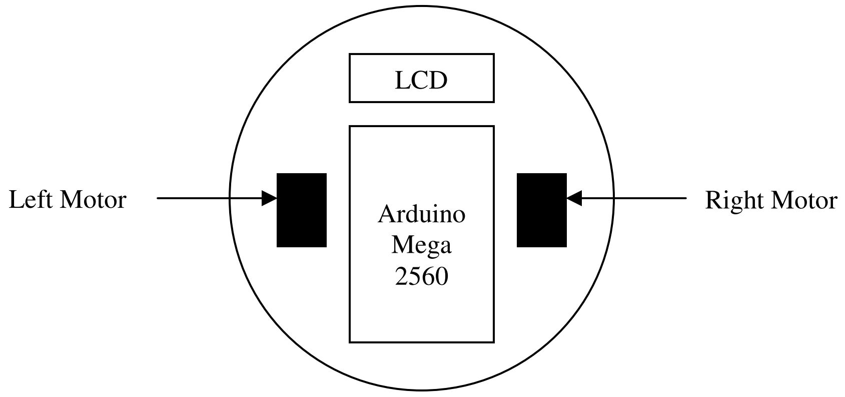
The following commands must be used to position the cursor at a specific location on the LCD screen:

```
LCD.write(0x7C);  // Put the LCD screen in command mode.
LCD.write(col + row*64 + 128);  // Place the cursor in the
position (row,col).
delay(10);        // Delay operation for 10 ms
```

The row and column of the cursor are denoted by the variables row and col respectively. The top and bottom rows of the LCD screen are described with the numbers 0 and 1 respectively. Each row of the LCD screen is 16 characters wide with each position denoted with the numbers 0 through 15 inclusive.

## Robot Technical Specifications

The diagram below provides a high level illustration of the robot for the purposes of describing its specifications:



The table below lists the pin numbers for all the devices connected to the Arduino platform and the possible operations for each pin.

| Device | Pin Number | Operation |
|---|---|---|
| Left motor | 2 | Write |
| Right motor | 4 | Write |
| Arduino Board LED | 13 | Write |
| LCD display | 18 | Write |

The table below illustrates common commands that can be used to manipulate the LCD screen:

| Command | Value (Hex) |
|---|---|
| Clear Screen | 0x01 |
| Turn off the display | 0x08 |
| Turn on the display | 0x0C |
| Scroll text to the left | 0x18 |
| Scroll text to the right | 0x1C |
| Turn off back light | 0x80 |
| Turn on back light | 0x9D |
| Command flag | 0xFE |

The table below illustrates the pulse width values required to manipulate the motors:

| Motor | Direction | Pulse Width Value |
|---|---|---|
| All Motors | Stop | 0 |
| Left Motor | Backward | 10 |
| Left Motor | Forward | 191.5 |
| Right Motor | Backward | 191.5 |
| Right Motor | Forward | 10 |

## Software Commenting Suggestions

The following templates are to be included in source code:

- File template block: Placed at the beginning of every file to indicate its purpose.
- Function template block: Placed above every function to describe its inputs, outputs and purpose.

**File template block:**

```
/*****************************************************************************
 *
 * Names:          John Smith  1234567
 *                 Jane Doe    4567890
 * Course Code:    SEG 4145
 * Lab Number:     1
 * File name:      filename.
 * Date:           January 1, 2013
 *
 *
 * Description
 * **************
 * A simple description of the file goes here
 *
 *
 *****************************************************************************
 */
```

## Function template block:

```
/*****************************************************************************
 *
 * Name
 **************
 * functionName
 *
 *
 * Description
 * *************
 * The description goes here
 *
 *
 * Parameters
 * *************
 * Name         Type         In/Out          Description
 *-----------   ----------   ---------------  ---------------
 *x            int          In              Description of x variable
 *y            char         In              Description of y variable
 *z            double       In              Description of z variable
 *
 *
 * Returns
 * *************
 * Name         Type         Description
 *-----------   ----------   ---------------
 *a            int          Description of a
 *
 *
 *****************************************************************************
 */
```