EMBSYS 110 Assignment 1

LED Pattern

1 Goal

The goal of this assignment to implement an LED pattern generator using object-oriented and statechart techniques. This approach allows the behaviors to be clearly specified and easily extensible.

2 Setup

1. Download the project compressed file (platform-stm32f401-nucleo_assignment_1_led.tgz) from:

   https://canvas.uw.edu/courses/1188665/files/folder/Assignments/Assignment%201%20LED

   Place the downloaded tgz file in **~/Projects/stm32**.

2. Backup your existing project folder, e.g.

   mv platform-stm32f401-nucleo platform-stm32f401-nucleo.bak

3. Decompress the tgz file with:

   tar xvzf platform-stm32f401-nucleo_assignment_1_led.tgz

   The project folder will be expanded to **~/Projects/stm32/platform-stm32f401-nucleo**.

4. Launch Eclipse. Hit F5 to refresh the project content.

   Or you can right-click on the project in Project Explorer and then click "Refresh".

5. Clean and rebuild the project. Download it to the board and make sure it runs.

   Note the USER LED no longer blinks. This is normal. Also you can remove the WIFI and IMU expansion boards (very carefully) to allow a better view of the USER LED.

   In minicom, you should see a new command named "led". Try out the new command "led on 0" or "led off". It does nothing now but it will start or stop an LED pattern once this assignment is completed successfully.

   ```
   22562 CONSOLE_UART2> led ?
   [Commands]
   test           Test function
   on             Start a pattern
   off            Stop a pattern
   ```

```
?              List commands
```

3 Tasks

1. All required tasks are marked with "// **Assignment 1**"
2. Design and add two new LED patterns in **src/UserLed/LedPattern.cpp**.

   You should understand the structure of the pattern table by referring to LedPattern.h. Two samples are provided for reference. Do not remove them. (They are pattern 0 and 1; your new ones will be pattern 2 and 3.)
3. Implement the UserLed active object (state-machine) in **src/UserLed/UserLed.cpp**. Hints are provided. Make sure you understand the design by referring to the statechart in src/UserLed/UserLed.pdf.
4. The statechart was created with the free tools named **UMLet**. You can download the latest version from:

   http://umlet.com/changes.htm

   The stand-alone version is more stable. It runs on Java, so it should work in your VM which has Java installed. The use of this tool is not required to complete this assignment.
5. Test your code with the console command:

   led on <pattern index from 0 to 3 > <0 for non-repeating; 1 for repeating>

   (e.g. "led on 2 1" to show pattern 2 repeatedly.)

   led off

   See how you can start a new pattern or stop it at any time. The design is extremely responsive, which is the essence of real-time systems.

4 Submission

The due date is 4/23 Monday 11:59pm. Please submit:

1. A zip file containing the source code in **src/UserLed**. If you have modified other folders you can include them as well.
2. A log file capturing the output of the UART console. You can turn on capturing in minicom by:

   sudo minicom ***-C log.txt***

   Your log should at least contain the output of the following commands:

   a) led on 2 0   (shows pattern 2 once)

    b) led on 3 1 (shows pattern 3 repeatedly for at least 3 cycles)
    c) led off (stops pattern 3)
3. Upload to the usual Canvas assignment upload location.