Test #1

17 april 2020

Note: it should go without saying that **plagiarism, in any form, will not be allowed**. Students are expected to provide their own answers to the questions. This form of exam allows for some liberty in preparing and writing your answers, which is equivalent to the liberty what students will find in their professional life: resources are available (books, the Internet, colleagues, ...) and due usage of these resources is left to the individual, based on her/his own ethics. Plagiarism (or any form of infringement of intellectual property) is not allowed, in business nor in academia.

All work submitted by the students in this exam may be subject to an additional oral presentation, in case doubts arise on the issues of originality or authenticity.

Part 1 – Essay questions

Answer the following question online, on the e-learning site. You may provide your answer in Portuguese.

Provide your comment on the paper “Why MISRA-C won't save your project”. Begin your answer by describing MISRA-C in your own words, how it can/should be used and its impact in embedded systems development. Proceed, then, to your comments on the paper above. (400 to 800 words)

Submit as a pdf file.

Part 2 – Code development

In the moodle web page, you will find a zip file with templates for the code. Use this as a start point for your answer.

Problem

Your job is to implement a temperature controller, to control a heating system. This controller has the following modes of operation:

- **Auto**: switches heating on and off when temperature is below or above a given threshold. This threshold is defined in the “Temperature set” mode.
- **Always ON**: forces heating always on, no matter the temperature.
- **Always OFF**: forces heating always off, no matter the temperature.
- **Temperature set**: allows for setting the temperature threshold. In this mode, heating is always off. The temperature is set in a range from 10 ºC to 30 ºC.

When the controller is switched on, the system goes to state “Temperature set”.

For the controller firmware, inputs are:

- Temperature: the temperature reading value, returned by a function readTemp(). readTemp() reads the temperature and return the temperature value as a 16 bit unsigned integer (uint16_t) in tenths of ºC, in a range from 0 ºC to 90 ºC (for example: TempRead() should return 143 for a temperature of 14.3 ºC).
- User input
  - User input has 4 press keys:

- **NEXT**: causes the system to change to the next mode, in the sequence defined above and in a circular fashion (1st to 2nd, 2nd to 3rd, ... , last to first).
- +: in Temperature Set mode, increases the threshold value. In Auto or Always OFF modes, forces heating output on for 5 s.
- -: in Temperature Set mode, decreases the threshold value. In Auto or Always ON modes, forces heating output off for 5 s.
- **Temp**: when pressed, displays the current temperature in the display

◦ the 4 keys are read in a bitmap by the function uint8_t keyRead() as follows:

Bit	8	7	6	5	4	3	2	1
Value	--	--	--	--	NEXT	+	-	Temp

The outputs are:

- Heating control: assumes ON of OFF values depending on the temperature reading. It is controlled by the function void setHeatingOn(bool value). value==true switches heating on, value==false sets heating off. bool getHeatingOn() returns the current mode of heating output.
- Display: a 16 char, alpha-numerical display. The display changes according to mode:

Mode	Display
Auto	displays mode, setpoint and current mode of heating Example: `AUTO T=25 ON`

Always ON, Always OFF	displays mode
	Example:
	`Always_ON`
	`Always_OFF`

Temperature set	displays mode, current value of setpoint and current mode of heating Example: `TSET T=25 OFF`

- Display is set by the function void setDisplay(char *).

Assignment

Consider that the functions defined above (readTemp(), setHeatingOn(), setDisplay(), ...) are developed by someone else. The functions for dealing with heating and user input are provided in the templates. You are provided with a template program that executes a simulation of the physical system.

Specify the state machine for this controller in UML or Harel Statecharts. Implement the state machine in a working C program, using the templates provided. The state machine should be totally included in the module heatingSM (files heatingSM.c and heatingSM.h). Your module heatingSM should include the function getCurrentStateName(), that writes the name of the current state to a buffer (see dummy example in the files).

The tester.c program contains a definition of macro called DEMO. This macro definition should be commented or removed from the code you submit.

Submit your code as a zip file, with the same structure as the zip file you received to start working.