KON309E Microcontroller Systems

Final Project: Temperature Controller

Subject:

You are expected to build a temperature controller that is similar to an industrial temperature controller. Such controllers are typically used in the temperature regulation of facilities. Some examples are cold storage units such as industrial refrigerators, hatching rooms for eggs and chicks, domestic heating and cooling and innumerable others. They have a display for showing a set-point, some buttons for adjusting it, a temperature measurement sensor input, and a control algorithm.

They are typically set as either a heating controller where they control a heater device, or a cooling controller, where they control a cooling element. In this application we will design a device which does both heating and cooling.

For examples see:

https://en.wikipedia.org/wiki/Thermostat for a general description, or

https://enda.com/automation/temperature-controllers/ for datasheets of how they are set and used.

Also search for w1209 temperature controller for a minimalist version that is simpler to understand.

Task:

The temperature controller will work as follows:

- Use the LM75A temperature sensor to sense the ambient temperature. Connect over the I^{2}C bus.
- Use two buttons to adjust the set point temperature (temperature reference).
- Two LEDs with different colors connected to two PWM output channels. (**Since we do not have heaters or refrigerators to control, we will mimic them using LEDs.**)
- Use the USB to TTL UART converter to describe what the system is doing at any given time.
- Use a constant sampling time, set by a timer ISR.
- Calculate the temperature and activate the correct heater or cooler outputs **in proportion to the temperature error**.
- One more LED must show if the control output is saturated (at max power). (In control applications, this is not desirable and means that the actuator is not able to cope.)

In the project you must demonstrate the following methods and competencies:

1. The setpoint temperature must start from 22 degC by default.
   - The measured temperature must be converted to degrees 'C' with a precision of 0.125degC.
2. Use different colored LEDs for representing the intensity of the heater or the intensity of the cooler.
   - The heater should preferably be represented by a red LED, and the cooler by a green or blue LED. If you do not have these colors, they can be the same color, but label them properly during the presentation.
   - The intensity of the cooler or heater is represented by the brightness of the LED. It must be adjusted through the duty ratio of the PWM.
   - Two PWM output channels of the same timer must be used.
   - Either the cooler LED must be ON or the heater. Both must not be ON at the same time.
   - Use PWM generated by a timer. PWM frequency must be 1kHz, and PWM resolution (ARR value) must be 100 steps. Set up the timer accordingly.

2. Use a baud rate of 115200, 8N1 configuration. Send clear messages as "Heating", "Cooling", PWM duty ratio, and current temperature.
   - If the set button is pressed, display the new setpoint.
3. The buttons increase or decrease the set point temperature respectively, by 0.5degC.
4. Buttons respond at each *release*.
5. Use timer to set the sampling time. The sampling time must be fixed and set to 0.5s.
6. The **gain of the controller** must be 5 PWM counts for every 0.125degC temperature error.
7. Your program must be based on a finite state machine (FSM). Draw the FSM using Drawio (https://app.diagrams.net/). The state names in the drawing must match the state names in the program. The double click routine must also be in the FSM.
   - The temperature control, setpoint adjustment and PC communications can be on different FSMs.
8. There must be no glitches in the operation. Every time a button is pressed, the controller must respond as expected.

Bonus:

A bonus will be awarded if you can configure your program to directly draw the temperature changes graphically on the PC using the "SerialPlot" program. In that case, you do not need to send text messages to the PC. The graphic plot must show 3 plots: the set point, the current heater intensity and the current cooler intensity. (Scale the values so that they fit nicely into the same graph - i.e., if the temperature is 25degC, but the intensity is 1000, then they will not fit nicely in the same graph).

Design essentials:

For this project, it is crucial that you get these items correctly before attempting the design:

1. Select the peripheral devices carefully! Decide on which pin for the button, which timer to control brightness and which timer for triggering the ADC. Decide on which ADC to use.
2. Document which peripheral is used for which purpose and how it is set up, in a list. For example, one list item may be: "TIM1 PWM CH1 is connected through pin PA8, to LED1 to show the intensity of the heater." etc.
3. Draw a circuit diagram.
4. Work on an algorithm that reads the button press correctly and without a glitch. At every press you should get a clean result.

Report contents:

Your report must be in a formal report format and contain the following:

1. The peripheral device assignment list. (See items 1, 2 in Design essentials)
2. FSM design. Show the state transition diagram that you save from Drawio application.
3. Your source code. Attach source code **only to the electronic submission** and not the paper submission. **Do not attach as a screenshot,** but as text, so that the TA's can replicate your code.
4. The report must be submitted in PDF format.
5. Upload your code after compressing it in ZIP format. (RAR format is proprietary and **will not be accepted**.)

Remember: Do not code in front of the computer, code on a piece of paper! This means:

1. Think and design your system,
2. Discuss with your group members on how to implement.
3. Write down your specifications and solutions.
4. Decide on which peripherals will do what and how they must be configured.
5. Only after that start coding.

If there is anything you do not understand, please contact the class assistants.