Real Time Systems using FreeRTOS – Final Project

Seat Heater Control System

Implementation of a seat heater control system for the front two seats in the car (driver seat and passenger seat).

Each seat shall consist of:

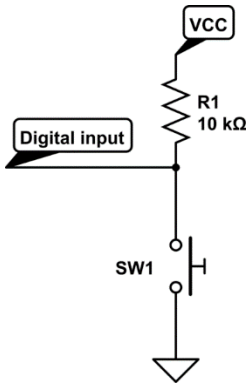
1. A button that is used to take the input level required to set the seat temperature.
2. A temperature sensor to monitor the temperature value.
3. A heating element to control the temperature value based on the desired level using a variable intensity power as input.
4. Shared screen between both seats to show the current state of the system.

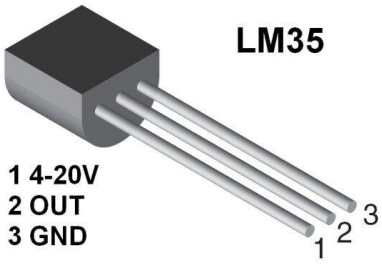
System features:

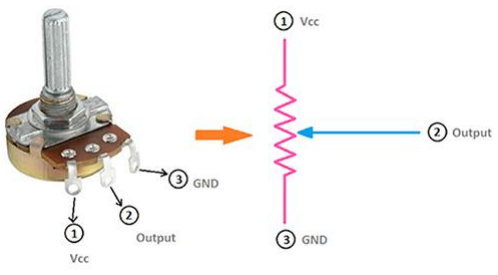
1. Heating level shall have one value of the following:
    a. Off: The feature is off, and the temperature is not controlled.
    b. Low: the desired temperature is 25°C.
    c. Medium: the desired temperature is 30°C.
    d. High: the desired temperature is 35°C.
2. The system shall control the temperature to be set within the range of the desired temperature +/- 2°C.
3. Diagnostics shall be present in case of failure of the temperature sensor.
    a. If the temperature sensor gives a value out of its range this is considered as a failure in the sensor.
    b. The range of the temperature sensor shall be 5°C-40°C.
    c. Such issues shall be logged in the memory according to the requirements.
4. All the data required to be presented to the user shall be shown on the screen of the car.
5. System contains 3 different buttons as shown in the images:
    a. 2 buttons in the car's middle console where each button is used to control one of the two seat heaters.
    b. 1 extra button in the steering wheel to control the driver seat heater to make the usage of the heater easier for the driver.

Functionality:

1. For each seat the user shall input the required level of heating (off, low, medium, or high) where initially off is set and each press goes one step further (from off to low, from low to medium, from medium to high, and from high to off once again).
2. The heater will be driven from a signal to control its intensity:
    a. If the current temperature is less than the desired temperature by 10°C or more the heater should be enabled with the high intensity.
    b. If the current temperature is less than the desired temperature by 5°C to 10°C the heater should be enabled with a medium intensity.
    c. If the current temperature is less than the desired temperature by 2°C to 5°C the heater should be enabled with low intensity.
    d. If the current temperature is more than the desired temperature the heater should be disabled.
    e. The heater shall be enabled once again if the temperature becomes less than the desired temperature by 2°C.
    f. Note that for testing purposes only, the heater level in this project is simulated to control an LED:
        i. The **green** color indicates **low** intensity.
        ii. The **blue** color indicates **medium** intensity.
        iii. The **cyan** color indicates **high** intensity.
3. The temperature sensor shall be connected to the ADC so that the current temperature is measured correctly.
    a. You can use an LM35 temperature sensor to measure the temperature.
        i. Refer to the datasheet of the sensor for its specifications.
    b. For the purpose of testing only, you can use a potentiometer connected to the ADC instead of the temperature sensor to take the temperature as input from the user with the following specifications:
        i. 0v-3.3v is mapped to the range 0°C-45°C.
        ii. Note that only the range 5°C-40°C shall be treated as the valid range.
4. The current temperature, the heating level, and the heater state should be displayed on the screen by sending it through the UART.
5. If a failure is detected in the temperature sensor (i.e., the reading is not within the range of 5°C to 40°C), the seat associated with that sensor will stop controlling the temperature (the heater will be disabled), and the corresponding red LED will turn on to inform the user of the sensor issue. When the temperature sensor returns to valid readings, the heater will be re-enabled, and the red LED will turn off.







Requirements:

1. It is required to implement the controller unit using Tiva C.
2. Software shall use FreeRTOS for handling different tasks and objects.
    a. The system shall be well-designed with at least **six tasks** (note: one function used for multiple tasks is considered as a single task).
    b. Task implementation shall be reused for two different instances (one for the driver seat and another for the passenger seat) if required.
    c. Shared resources must be identified and mutexes should be used to provide exclusive access to them.
    d. Event-based tasks shall be identified and managed using event flags or semaphores.
    e. Data sharing between different tasks must be handled properly without any loss of data.
    f. Responsiveness to buttons shall be as high as possible, and minimizing CPU load is a priority. Therefore, edge-triggered interrupts shall be used instead of polling.
3. GPIO, UART, GPTM, and ADC modules shall be implemented as part of the MCAL modules and used in the software.
4. Diagnostics shall be implemented where each of the following will be stored in the RAM.
    a. The failure along with the timestamp (using GPTM) at which the failure occurred.
    b. The last heating level set by the user (off, low, medium, or high) with its timestamp shall be recorded.
5. **Manual runtime measurements** shall be made using the **GPTM** for the system to calculate the following parameters with the highest possible resolution (0.1 ms accuracy for our GPTM implementation):
    a. Execution time for each task.
    b. CPU load.
    c. Resource lock time per task for each resource (using trace hook macros so please search for the appropriate trace hook to use).

    You can use the FreeRTOS runtime statistics feature to validate your readings, but it is not accepted as part of the project delivery.

Hardware Components:

Here is a summary of the hardware components to be used in the project:

Each seat, whether passenger or driver, includes:

- A temperature sensor (simulated as a potentiometer).
- A red LED for error reporting.
- A heater (simulated with two LEDs: green and blue).
- A button for controlling the heating level.

Additionally, the driver has an extra button on the steering wheel.

Testing Scenario:

1. The passenger sets the heating level to high while the temperature is 10°C.
2. Initially the heater will be driven with high intensity.
3. When the temperature reaches 25°C the heater will be driven with medium intensity.
4. When the temperature reaches 30°C the heater will be driven with low intensity.
5. When the temperature reaches 35°C or more the heater will be disabled.
6. Whenever the temperature goes down to 33°C, the heater will be reenabled with low intensity.
7. The required data will be sent by UART to be displayed on the screen.
8. If the temperature was sensed to be with a value less than 5°C or more than 40°C the controlling of the temperature shall be disabled, and the user shall be informed by enabling the LED according to the requirements.

Bonus:

Implement non-volatile memory for the diagnostics part:

1. The last heating level set by the user (off, low, medium, or high) with its timestamp shall be saved in the non-volatile memory.
2. The diagnostics issues logged in the system shall be saved in the non-volatile memory as well.

Deliverables:

1. Application project including the following:
    a. Application tasks source code.
    b. FreeRTOS configuration file.
    c. MCAL modules source code.
    d. Tested and working elf or hex.
2. Simso simulation project
    a. Please round up any readings to the nearest millisecond, and if you encounter a reading with a zero value (less than timer resolution), simulate it as 1 millisecond.
    b. Please ensure that any tasks exceeding their deadlines, as indicated by the red arrow in the Simso simulation, are addressed. Aim to deliver a simulation that is free of any missed deadlines as much as possible.
3. A PDF document contains the following:
    a. A diagram for the system design containing all task details, including task description, type (periodic, init, or event-based), periodicity for periodic tasks, and events that tasks set or wait for.
    b. A list of shared resources, indicating which entities share them and the method used for their exclusive access.
    c. Screenshots for the output of the system (UART messages).
    d. Run time measurement results.
    e. Simso simulation results.

Thank You

Edges For Training Team