The Roux Institute
Northeastern University

CS Programming Embedded Systems
Fall 2024

Homework 02 **Assigned:** Fri 20 SEP 2024
Coding a Controller [100 points] **Due:** Sun 29 SEP 2024

Introduction:

One of the more popular design models we reviewed is the Model-View-Controller (MVC) software design pattern. Basically, MVC defines that each section of your code has a specific purpose, and each of those purposes are different. One section of code might hold all the data for a system; another section might be responsible for receiving and sending signals to sensors and devices; while a third section might be the user interface that communicates the status of the system. MVC is a way to organize your code's core functions into their own discrete bins, making your system design easier to consider when expanding with new options, incorporating your system within a larger or parallel system, or sharing your system components with others.

Problem:

We want to create an application that regularly receives input from a sensor. Upon receipt of this information, our application should check to see if anything reported by the sensor has changed the known world we are monitoring. This set of actions would be equivalent to controllers located in automobiles, home automation, and even traffic light systems. If the application discovers anything different from the sensor, the application needs to act and update the known world that is being monitored.

Some features to be included in the application:

- The system should report the current temperature when initially started.
- The system should default to a desired default temperature for operation of 72 degrees Fahrenheit.
- The system should report the new temperature when the sensor reports a change in temperature.
- The system should allow the user to adjust the desired temperature by entering into a curser prompt the code **`t nn`** where the `t` command uses the numerical value of `nn` to replace the desired temperature, in Fahrenheit, with `nn`.
- When the desired temperature is adjusted, the system should report the current temperature.

A sample run of the application might appear to the user as (since the sensor is random, your output may be different):

```
Welcome to Den
Set Temperature: 72F
Current Temperature:  72F
Current Temperature:  71F
Current Temperature:  70F
Current Temperature:  69F
Set Temperature: 74F
Current Temperature:  69F
Current Temperature:  70F
Current Temperature:  71F
Current Temperature:  72F
Current Temperature:  73F
Set Temperature: 70F
Current Temperature:  73F
Current Temperature:  72F
```

Your application should run continuously, though you can offer an option to shutdown and exit by entering the command letter `x`.

Solution:

You are to create an embedded systems design of the above-described thermostat and use C to code a simulation of a working thermostat. Your model and implementation should (to the greatest extent possible) include the requirements above, though you are free to make modifications that are included in your design and described as variations to the requirements. You must use the simulated thermostat function provided in GitHub for HW2, which you can build on your Raspberry Pi. To establish a Git connection with the Class Repository, issue the following command in a development folder on your Raspberry Pi:

```
git clone https://github.khoury.northeastern.edu/sav/CS7680_ClassRepo.git
```

Then, once your connection is established between your Pi and the Repo, collect the code for this assignment by doing a **`git pull`** on the Class_Repo and copy the files required (`fakenews.o`, `fakenews.h`, `makefile`) from the Repo folder into your working development folder from where you will upload your solutions to your private repository. Once moved, run the **`make`** command on the HW2 files acquired in that folder.

Any variations in the design should enhance the final solution, be implemented in the software, and not decrease the proposed features.

Tasks:

Task A [30 points]: Develop a design model that is clear and captures the requirements and/or adjusted requirements.

Task B [50 points]: Develop a controller program called **`den`** that uses the `fakenews()` function as a simulated sensor data source.

Task C [20 points]: Write a two-page maximum, single-spaced paper that serves as the narrative to explain your solution.

Submission:

- Your completed submission should be uploaded to GitHub by the due date, which is scheduled for 11:59pm ET that day since solutions will be distributed soon after.
- We expect that you will study with classmates and often work out problem solutions together, but *you must write up your own solutions, in your own words*. **Cheating will**

not be tolerated. Professors and TAs will be available to answer questions but will not do your homework for you. One of our course goals is to teach you how to think on your own and solve your own problems using your resources. Cut-and-paste from Google or ChatGPT will be considered plagiarism. If you consult a reference, be sure to indicate such in a reference section as part of your submission.

- We require that all submissions be neat and organized. All document submissions should be in Adobe PDF format. **There will be point deductions if the submission is not neat** (is disordered, difficult to read, scanned upside down, etc.).
- To achieve full credit, **show INTERMEDIATE steps, if applicable,** leading to your answers throughout.

Reference Sources:

Herskowitz, Daniel, *Medium*, Website viewed 17 SEP 2024, https://medium.com/@beyondnumbers/defining-the-problem-thermostat-design-part-1-2c78eb75a40c

Codecademy, Website viewed 20 SEP 2024, https://www.codecademy.com/article/mvc