Category One: Software Design and Engineering

What does the code do?
The artifact chosen for this category is from a course regarding emerging system architecture and technologies. The code in the project was intended to create and prototype an embedded system smart thermostat. The thermostat was intended to sense a current temperature and compare it to a setpoint temperature. If the current temperature were to be below the setpoint, an LED would light (simulating a heating system turning on). The setpoint was controllable by two buttons on the TI board (one to increase the setpoint and one to decrease).

Focus on the features and functions of the existing code.
In the gpio interrupt file you will see that the necessary peripherals are imported (being I2C for temperature, UART for data sending simulation, and a Timer). There are global variables declared for use in each.

There is additionally a callback function for the timer and one for each of the two buttons. These callbacks change a flag marker in the case of the timer period ending, or a button being pushed to change the current set point.

Next there are initialization functions for each of the aforementioned peripherals. Generally, in each, a driver is initialized, configured, and opened. I2C specifically will involve scanning possible sensors for the given board and setting the sensor address as target addresses.

Certain peripherals like I2C require some additional functions such as the readTemp function. This allows the current temp to be extracted from sensor data.

Throughout the main thread a common denominator is used through the timer period for different event times. Configurations and callbacks are set, and there is a main while loop. In the main loop the flags mentioned in use with the callback functions are checked periodically, and appropriate action is taken if flags are true.

Analyze existing code using relevant code review criteria to support clearly stated findings.

Structure
There are some residual stubs or test routines as seen in the code. These would be cleaned up through enhancements.

There are some variables that you might call magic number constants in lines 316-318. These elapsed time variables are representing microseconds as they are used in comparison to the timer period, which by the header file specifications are units of microseconds. Additionally, the output which occurs every second needs to be cleaned up (and that is why heat is a confusing variable in line 400). Otherwise, providing the appropriate documentation with appropriate naming on these variables will aid in clarity.

Documentation
There needs to be a little greater consistency throughout the commenting in the code, which will be addressed with enhancements. The code is, for the most part, adequately documented with information, though some could be added.

Variables
Some self-description could be added to some variables such as with the elapsed time variables by adding in microseconds, or with some of the method names such as changing the button callbacks from button callback increase or decrease to button callback increase or decrease temperature or making read temp read current temperature. These aspects will provide greater clarity and readability while decreasing the need for unnecessary documentation.

Otherwise, all variables are used.

Loops and Branches
Within the main while loop, to ensure all cases are covered properly and avoid unnecessary checks in if-elseif blocks, the condition where the current temp is equal to the set point should be moved to an else or default clause (around line 385).

Otherwise loops and branches are complete, properly nested, and indexed properly.

Security
Currently, the device state is not changed upon termination, and could be altered to reconfigure the board LEDs upon termination.

Efficiency
The efficiency of the code could possibly be increased by the implementation of a task scheduler in place of the conditional checks used for the individual tasks in the while loop currently.

Explain practical enhancements aligned with analysis findings in a clear and organized manner.

Overall enhancement plans include adding more complete functionality in terms of a smart thermometer that will enable the reverse to occur, where an LED will indicate if a current sensed temperature is above the setpoint (simulating an AC aspect).

Structurally, removing unnecessary test code and clearing up variable naming conventions for detail and clarity will aid in readability. Moreover, greater self-description is needed throughout certain functions. This will further be aided by consistent and adequate documentation.

Some work is necessary on some of the loops and branches to ensure all cases are properly covered.

Lastly, review of tasks scheduler possibilities will increase efficiency.

The specific course outcomes that these enhancements support include:

“Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry specific goals.”

“Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.”

Skills related to embedded systems and coding practices will be used to implement these changes. Additionally, knowledge of peripherals will be implemented. In order to provide a prototype solution, the culmination of these factors accomplish a specific goal with regards to industry standards such as sending data to servers via WIFI. WIFI solutions are weighed also, involving managing tradeoffs.

Category Two: Algorithms and Data Structures
Provide a clear, complete description of existing code **functionality.**

What does the code do?

The artifact selected for this category is a course planner program from a data structures and algorithms course. This project was centered around a client developing a program to aid academic advisors in assisting students. The program loads a file of course information (in this case computer science courses) and allows advisors to print schedules or search for a specific course. This project was ultimately about determining appropriate data structures for a client's needs.

Focus on the features and functions of the existing code.

In the course planner file a course structure is defined to hold course information (such as the course number, name, and any pre-requisites). A hash table is used to store course information, and a class defines the structure.

There are methods such as a constructor to initialize the structure and resize the table and hash method that calculates hash values based upon that table size.

For functionality, there is an insert course function, which creates a key for a course and inserts it into the hash table accordingly with pointers. There is a print all courses function that iterates through the nodes, printing the course information until the last course in the structure is reached. There is the ability to search for a specific course by course number (seen as the unique identifier above), and to print course information for a user. Lastly, there is a method to load a csv file of course information for use to the hash table.

The main method creates some variables and displays a menu of options that will repeat until the user chooses to exit the program. A choice is read in, and a switch case is used to handle the menu options of choices. Menu options include the ability to load an alternative data structure by providing a path to a CSV file, print the course list, or print a specific course by providing a course number.

There is additionally a csv parser module that allows for uneven header and row lengths.

Analyze existing code using relevant code review criteria to support clearly stated findings.

Structure
The code completely implements the design proposed by the assignment for necessary functionality but could be enhanced to allow for greater searching.

The code is consistent in structure, style, and formatting.

Some blocks of code, for example the code that displays the menu of options or the choices switch case handling in the main loop, could be moved to their own functions that could be called to display menu and handle choice.

Using a hash table data structure is generally efficient and provides fast data retrieval, but alternatives could be reviewed to ensure maximum efficiency.

Documentation
The code is adequately and consistently commented, but any additional methods created should continue following this pattern.

Variables
All variables have meaningful names with consistent naming conventions (using camel case). The method naming could be more self-descriptive and consistent in casing and style. For example, the method insert inserts a course into the hash table but insert is not descriptive as to what is being inserted as is. This is also true for search and hash (arguably). More appropriate descriptive names could be insert a course and search for a course or search by course number.

Loops and Branches
All cases are covered with most common being tested first. A lot of the conditionals are to check for the presence of something such as prerequisites as not every course will have them.

Security
When allowing a user to enter a file to load in menu option and switch case 1, there is a lack of validation to ensure that the existence of the file is first checked for before attempting to load it in line 285.

Further input validation could be added where user input is accepted for aspects such as type, or length is lacking (when accepting a csv path or a course key for example).

Lastly, memory leaks need to be addressed when allocating memory such as in line 261 creating a new course table. Declarations such as these should be deallocated.

Explain practical enhancements aligned with analysis findings in a clear and organized manner.

Overall, I plan to review the efficiency of the program. A data structure was initially selected based upon runtime and usage evaluations. This was derived from pseudocode for various structures (vectors, hash tables, trees). Individual structures will be reviewed and reconsidered for efficiency, as well as the code responsible for the chosen structure. This could include reviewing the total cost of the algorithm used with the selected structure as well. Additionally, greater functionality could be implemented.

Structurally, reworking the amount of code contained in the main function and moving aspects such as the menu and handling to their own functions will enhance readability and code quality.

Addressing the necessary variable and method naming for descriptiveness and consistency will further improve readability and decrease the need for some unnecessary documentation.

Lastly, addressing concerns surrounding input validation and memory allocation and deallocation will lead to a more secure end product that conforms to standards.

The specific course outcomes that these enhancements support include:

"Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices"

To align with the outcome, creating algorithms with selected data structures lends to solving problems using algorithmic principles, while more importantly reviewing efficiency and cost will involve reviewing and weight trade-offs (such as worst cases, best cases, total cost, etc.).

Category Three: Databases

Provide a clear, complete description of existing code **functionality**.

What does the code do?

The artifact chosen for this category is one from a full stack development class. The client in the scenario had requested development of a travel web application. . The application is generally intended to allow customers to book travel packages to various destinations. It currently utilizes a MongoDB database to store travel package information, with a REST API used to basic database functionality (allowing get, post, put, and delete functionality).

Focus on the features and functions of the existing code.

The travlr.js file under models provides a trip schema for the necessary attributes of a trip in the database.

The db.js file builds the connection string and sets the connection timeout, as well as monitors connection events with listeners. This allows for graceful shutdowns in the event of shutdowns caused by signal or unexpected termination issues.

The seed.js file takes the database connection exported from the db.js file, and the trip model exported from the travel.js file. Seed data is read from a Json file, which is parsed. The parsed data is inserted into the database, and the connection is closed.

In the index.js file, routes are defined for the trips API endpoint (which is trips).

Within the trips.js file, methods to retrieve a list of all trips (trips list), to retrieve a single trip by trip code (trips find by code), to add a trip (trips add trip), and to update a trip (trips update trip) are defined.

Analyze existing code using relevant code review criteria to support clearly stated findings.

Documentation
The code shown throughout the files could benefit from more adequate documentation

Variables
The model for a trip uses strings for all aspects of trip attributes, and this is due to the fact that they can be entered in a form.

Security
All should be left in the correct state upon termination due to graceful shutdowns.

Where room for error exists, there are conditional checks that handle and assign appropriate status. Default cases could be added to the if else statements throughout the trips.js file.

Explain practical enhancements aligned with analysis findings in a clear and organized manner.

At the time of the project, only certain functionality was required when working with database entities. I plan on adding more advanced query concepts which will in turn allow for better searches and provide more useful results to users.

Adding more documentation to the files used throughout, especially in a larger project like this one where there are many files, will aid in readability and future maintainability.

Adding additional clauses and defaults to the present branches will ensure all cases are covered ensures best practices are being followed to allow handling under unpredicted situations.

Lastly, there is a level of user authentication surrounding some methods (such as adding trip listing information to the database) that needs to be addressed.

The specific course outcomes that these enhancements support include:

“Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.”

“Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources.”

The aforementioned necessary underlying fixes will promote security, creating user authentication and authorization to align with a security mindset. Utilizing databases, a MongoDB interface (Compass), and through testing (with tools or otherwise) a safer and more efficient solution will be developed.