SAVEETHA SCHOOL OF ENGINEERING

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

CHENNAI-602105

SIMULATING REAL-TIME TASK SCHEDULING FOR EMBEDDED SYSTEMS

A CAPSTONE PROJECT REPORT

Submitted in the partial fulfillment for the completion of the course

CSA0499 OPERATING SYSTEMS FOR VIRTUALIZATION

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

A.NIHAL UR RAHMAN(192111024)

Under the Supervision of

Dr. Jegatheesan A

FEBRAURY 2025

DECLARATION

I **A.NIHAL UR RAHMAN(192111024)** student of **Dr. Jegatheesan A** of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha School of Engineering, Chennai, hereby declare that the work presented in this Capstone Project Work entitled "**Simulating Real-Time Task Scheduling For Embedded Systems**" is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics.

DATE: 27/02/2025

PLACE: SIMATS,Chennai

CERTIFICATE

This is to certify that the project entitled " **Simulating Real-Time Task Scheduling For Embedded Systems** " submitted by **A.NIHAL UR RAHMAN** has been carried out under my supervision. The project has been submitted as per the requirements in the current semester.

Supervisor

Dr. Jegatheesan A

1. ABSTRACT

Real-time task scheduling is a fundamental aspect of embedded systems, where timely execution of tasks is crucial to ensure system reliability, responsiveness, and performance. Embedded systems often operate under strict timing constraints and limited resources, requiring an efficient and predictable scheduling mechanism to manage task execution. This paper presents a simulation-based approach to real-time task scheduling for embedded systems, focusing on the evaluation and comparison of different scheduling algorithms. We investigate the effectiveness of classical scheduling techniques, such as Rate-Monotonic Scheduling (RMS), Earliest Deadline First (EDF), and Least Laxity First (LLF), and evaluate their performance in a simulated embedded environment. The proposed simulation framework takes into account a variety of factors that influence real-time scheduling, including task priorities, system resource allocation, workload characteristics, and environmental variables such as power consumption and interrupts.

Embedded systems are typically characterized by stringent timing requirements, where tasks need to be executed within specified deadlines to avoid catastrophic system failure. The primary challenge in real-time systems is ensuring that tasks meet their deadlines while optimizing the use of available resources. Real-time scheduling algorithms are designed to guarantee that all tasks are executed within their deadlines and provide a predictable response to external events. However, different algorithms have distinct advantages and trade-offs, making it important to select the most suitable one for a particular application.
Rate-Monotonic Scheduling (RMS) is a fixed-priority scheduling algorithm where tasks are assigned priorities based on their periodicity. The task with the shortest period receives the highest priority. RMS is simple to implement and guarantees optimal scheduling for systems with periodic tasks, assuming that the system's utilization is below a certain threshold. However, RMS may not be suitable for systems with a high degree of variability in task arrival times or for aperiodic tasks.

Earliest Deadline First (EDF) is a dynamic-priority scheduling algorithm that assigns priorities based on the deadlines of tasks. The task with the earliest deadline receives the highest priority, making EDF more adaptable to varying task schedules. EDF has been proven to be optimal in the sense that if any schedule can meet all task deadlines, EDF will schedule those tasks successfully. However, EDF has its limitations, such as increased computational overhead and the potential for priority inversion in systems with nested interrupts.
Least Laxity First (LLF) is another dynamic-priority scheduling algorithm where tasks are prioritized based on their laxity

2. TABLE OF CONTENTS

3. ACKNOWLEDGEMENT

I take immense pleasure in expressing my sincere gratitude to the honorable chancellor, **Dr. N. M. Veeraiyan** for being a source of inspiration. I take this opportunity to express my gratitude to the Vice Chancellor **Dr. S. Suresh Kumar** for providing all the facilities to complete this research work successfully.

I sincerely thank the Director of Academics, **Dr. Deepak Nallasamy** for his visionary thoughts and support. My special thanks to the Director of SIMATS Engineering **Dr. Ramya Deepak** for being a constant source of inspiration and for providing excellent research ambience.

I would like to express my sincere gratitude to my research supervisor **Dr. A. Jegatheesan**, Professor, Department of Computer and Engineering, Saveetha School of Engineering College, for motivating around all obstacles and for being such an outstanding mentor.

My special thanks to my mentor **Dr. HEMAVATHI** for valuable support and guidance. I would like to thank HOD, all staff members, members of research department who were always there at need of the hour and provided the facilities, which I required for the completion of my thesis.

I would like to express my profound gratitude to my family members and friends for their encouragement and support. I would like to thank God for not letting me down at the time of crisis and guided me on right direction.

4. INTRODUCTION

Embedded systems are integral components in a wide range of modern technologies, including automotive control systems, medical devices, industrial automation, robotics, and consumer electronics. These systems are often characterized by strict performance requirements, including real-time constraints that demand tasks be completed within predefined deadlines. For embedded systems to function effectively, real-time task scheduling plays a critical role in ensuring that all tasks meet their deadlines while optimizing the system's resource utilization. Failure to meet these timing constraints could lead to catastrophic system failures, affecting the safety, efficiency, and reliability of the application.

Real-time task scheduling refers to the process of determining the order in which tasks or processes should be executed in a way that guarantees their timely completion. The scheduling strategy is of paramount importance because embedded systems typically operate under resource constraints, such as limited processing power, memory, and energy. In addition, tasks in real-time systems often have varying periods, priorities, and execution times, further complicating the scheduling process. The goal of real-time scheduling is to ensure that the system operates within the specified constraints, with tasks completed by their deadlines while maintaining high efficiency in the utilization of available resources.

There are several real-time scheduling algorithms that have been proposed for use in embedded systems, each with its own strengths and limitations. One of the most well-known algorithms is Rate-Monotonic Scheduling (RMS), a fixed-priority scheduling algorithm where tasks are prioritized based on their periodicity. While RMS is optimal for systems with periodic tasks, it may not perform well in dynamic environments with a mix of periodic and aperiodic tasks. Earliest Deadline First (EDF), on the other hand, is a dynamic priority algorithm that assigns priorities based on the task’s deadline, making it more flexible in handling a diverse set of tasks. EDF is optimal in terms of deadline feasibility, but its overhead and complexity may pose challenges in resource-limited systems. Another algorithm, Least Laxity First (LLF), dynamically adjusts task priorities based on the remaining time until deadlines and the remaining execution time, offering flexibility in certain real-time systems. However, LLF's computational complexity can make it unsuitable for systems with tight resource constraints.

5. PROBLEM IDENTIFICATION AND ANALYSIS

Description of the Problem:

Embedded systems are designed to perform specific tasks with high reliability and within strict timing constraints, making real-time task scheduling crucial. The primary challenge in embedded systems lies in effectively managing multiple tasks that require resources—such as processing power, memory, and energy—while ensuring that each task completes within its specified deadline. As these systems are frequently deployed in environments where failure to meet deadlines could result in catastrophic consequences, the scheduling of tasks becomes a critical design issue.

Evidence of the Problem:

The evidence of scheduling problems in these critical applications can be observed in real-world failures, where scheduling algorithms either fail to meet deadlines or lead to inefficient resource usage. These failures often arise due to the inappropriate selection of scheduling algorithms that do not account for system limitations or workload complexity, resulting in missed deadlines or unnecessary system overhead.

Stakeholders:

Several key stakeholders are affected by the issues related to real-time task scheduling in embedded systems:

- **System Designers:** The engineers responsible for the design of the embedded system need to choose and implement the right task scheduling algorithms to ensure the system meets performance and real-time constraints
- **End Users:** For users of embedded systems, whether they are consumers, manufacturers, or operators, reliable performance and timely task execution are critical.
- **Regulatory Authorities:** In industries such as healthcare and automotive, regulatory bodies enforce standards that mandate the reliability and performance of embedded systems.

Supporting Data/Research:

The importance of real-time task scheduling in embedded systems is well-supported by research and case studies across different industries. A number of studies have been conducted to analyze the performance of different scheduling algorithms under varying conditions.

6. SOLUTION DESIGN AND IMPLEMENTATION

Development and Design Process: The development and design process for real-time task scheduling in embedded systems typically follows a structured approach:

- **Requirement Analysis**: Identify system requirements such as task periodicity, deadlines, and resource constraints (CPU, memory, energy).
- **Algorithm Selection**: Choose appropriate scheduling algorithms based on task characteristics, such as RMS, EDF, or LLF.
- **System Modeling and Simulation**: Model the embedded system environment, including tasks, resources, and priorities. Simulate the system to evaluate the performance of selected scheduling algorithms.

Tools and Technologies Used:

- **Simulation Tools:** MATLAB/Simulink, ModelSim, and custom-built simulators
- **Embedded Development Tools:** C/C++ programming environments, Eclipse or Keil.
- **Operating Systems:** FreeRTOS or VxWorks for task scheduling and resource management.

Engineering Standards Applied

- **ISO 26262:** Automotive safety standard to ensure functional safety in embedded systems, particularly for critical tasks.
- **IEC 61508:** Standard for functional safety in industrial applications, ensuring reliability in real-time embedded systems.
- **MISRA C:** Coding guidelines for safety-critical embedded systems to ensure high-quality, maintainable code in real-time applications.
- **RTOS Guidelines:** Conformance with real-time operating system standards to ensure deterministic task scheduling and response times.

7. RESULTS AND RECOMMENDATIONS

Evaluation of Results:

The evaluation of the results from the real-time task scheduling simulation focuses on comparing the performance of various scheduling algorithms (RMS, EDF, and LLF) in meeting deadlines, optimizing resource utilization, and maintaining system stability under different workload scenarios.

- **Deadline Satisfaction**: The EDF algorithm consistently performed well in meeting deadlines, especially in dynamic environments with a mix of periodic and aperiodic tasks. RMS, while reliable in systems with fixed periodic tasks, showed limitations in systems with unpredictable workloads.
- **Resource Utilization**: RMS exhibited efficient use of CPU resources under predictable task sets, but EDF offered superior adaptability when task arrival times varied.
- **System Stability**: LLF demonstrated potential for instability, especially in systems with high task variability due to its complex recalculation of laxity.

Challenges Encountered:

Several challenges were encountered during the design and simulation of real-time task scheduling:

- **Computational Complexity**: Algorithms like LLF require continuous recalculation of task laxity, which can incur significant computational overhead in resource-constrained systems.
- **System Overhead**: Dynamic priority scheduling (like EDF) increases system overhead due to frequent task priority changes, making it unsuitable for systems with limited processing power.
- **Task Variation**: Handling systems with highly dynamic or unpredictable task sets posed difficulties in accurately predicting system behavior and ensuring deadline satisfaction.
- **Integration with Hardware**: Integrating the software simulation with actual embedded hardware for testing real-time constraints led to discrepancies in performance due to hardware limitations.

Possible Improvements:

- **Hybrid Scheduling Algorithms**: A hybrid approach combining RMS for periodic tasks and EDF or LLF for aperiodic tasks could provide a better balance between predictability and flexibility in scheduling.

- **Energy-Aware Scheduling**: Incorporating energy-efficient scheduling strategies could improve the overall system performance, especially in battery-operated embedded systems.
- **Real-Time Adaptive Scheduling**: Implementing adaptive scheduling that can adjust priorities based on the system's real-time performance (e.g., resource usage, power levels) could help overcome some of the limitations of static scheduling algorithms.
- **Optimization of Computational Overhead**: Reducing the recalculation frequency in dynamic algorithms like LLF could help minimize the computational burden, making them more feasible for resource-constrained environments.

Recommendations:

- **Hybrid Approach**: We recommend adopting hybrid scheduling algorithms that combine the strengths of RMS and EDF, depending on the task type (periodic vs. aperiodic), to achieve better performance in diverse real-time embedded systems.
- **Focus on Resource Constraints**: It is essential to tailor the scheduling algorithm to the specific resource constraints (e.g., CPU, memory, power) of the embedded system to avoid excessive overhead and ensure efficient operation.
- **Energy Efficiency**: Future work should prioritize the integration of energy-efficient task scheduling methods to extend the battery life of embedded systems in power-critical applications.
- **Continuous Testing and Validation**: Conduct thorough testing in both simulated and real hardware environments to ensure the scheduling algorithms meet the real-time performance and reliability requirements under varying workloads.
- **Advanced Real-Time Operating Systems (RTOS)**: Utilizing advanced RTOS with built-in support for real-time task management can simplify scheduling implementation and improve the overall system's reliability and performance.

8. REFLECTION ON LEARNING AND PERSONAL DEVELOPMENT

Throughout the process of designing, simulating, and evaluating real-time task scheduling for embedded systems, I have gained valuable insights into both the technical and personal aspects of system design and problem-solving. This experience has contributed significantly to my understanding of real-time systems, task scheduling algorithms, and how theoretical concepts are applied to practical, resource-constrained environments.

1. Key Learning Outcomes

- **Real-Time Scheduling Knowledge:** I gained a deeper understanding of real-time scheduling algorithms (RMS, EDF, LLF) and their advantages and drawbacks in embedded systems.
- **Simulation and Evaluation:** I learned how to simulate real-time task scheduling in embedded systems, evaluating algorithm performance under various workload conditions.
- **Resource Management:** I developed an understanding of managing critical system resources (CPU, memory, energy) effectively while ensuring deadlines are met.
- **System Optimization:** The project enhanced my ability to identify areas for optimization in embedded systems to meet performance and real-time constraints efficiently.

Technical Skills

One of the most significant outcomes of the project was the enhancement of my technical proficiency in both software and hardware aspects of real-time scheduling. The key skills acquired include:

- **Programming:** Improved my proficiency in C/C++ and embedded system development.
- **Simulation Tools:** Gained experience using MATLAB/Simulink for system modeling and task scheduling simulations.
- **Embedded System Design:** Gained hands-on experience with embedded development environments and tools like FreeRTOS, Keil, and Eclipse IDE.
- **Real-Time Operating Systems (RTOS):** Increased my knowledge of using RTOS for managing tasks, prioritization, and ensuring real-time constraints are met.
- **System Testing and Debugging:** Enhanced my skills in debugging embedded systems and testing for performance under real-time conditions.

Problem-Solving and Critical Thinking

This project required me to apply critical thinking to solve complex issues related to real-time task scheduling. I had to analyze and select the most appropriate scheduling algorithm based on the task set, syst em constraints, and performance goals. I also learned to evaluate and address system inefficiencies, such as excessive computational overhead in dynamic scheduling algorithms, and found ways to optimize the scheduling approach. Overall, the project reinforced my ability to break down large problems, assess potential solutions, and implement practical approaches to meet system requirements.

2. Challenges Encountered and Overcome

- **Complexity of Dynamic Scheduling:** Implementing dynamic priority scheduling algorithms, such as EDF, posed a challenge due to the frequent recalculations of task priorities. This added complexity resulted in increased system overhead and performance issues in resource-constrained systems. To overcome this, I researched optimization techniques, such as reducing recalculation frequency, and implemented a hybrid approach that combined static and dynamic algorithms.
- **Resource Constraints:** The limited processing power and memory of embedded systems introduced challenges in running resource-intensive scheduling algorithms. By analyzing the performance of each algorithm in different conditions, I was able to identify the most efficient scheduling strategy for each scenario, optimizing resource usage while maintaining deadlines.
- **Integration with Hardware:** Testing real-time scheduling on actual hardware was more challenging than expected due to hardware limitations, which led to discrepancies in performance. By carefully analyzing the results from simulations and conducting iterative testing, I improved the reliability of the system and minimized these discrepancies.

Collaboration and Communication

Throughout the project, I communicated regularly with mentors and peers to clarify doubts, troubleshoot issues, and receive feedback. While the project was primarily self-directed, collaboration with others provided me with new perspectives on problem-solving and algorithm design. Additionally, presenting my findings and simulation results helped me hone my ability to communicate complex technical concepts in a clear and understandable way, making it easier to convey my analysis and decisions to a broader audience.

3. Application of Engineering Standards

- **ISO 26262 and IEC 61508**: These standards are particularly important in safety-critical embedded systems, such as automotive and industrial applications. Understanding these standards has allowed me to design the task scheduling approach with safety and reliability in mind.
- **MISRA C**: I followed MISRA C coding guidelines to ensure the development of high-quality, maintainable, and error-free code, which is essential in embedded systems development.
- **Real-Time System Guidelines**: By applying best practices for real-time systems and RTOS, I ensured that the system met strict timing constraints and operated efficiently under different load conditions.

By following these engineering principles, I ensured that my scheduling approach aligned with **industry best practices, improving its applicability to real-world systems**.

4. Insights into the Industry

Working on real-time task scheduling has provided me with insights into the challenges faced by industries that rely on embedded systems, such as automotive, healthcare, and industrial automation. These sectors demand high reliability, low power consumption, and precise timing, which makes task scheduling algorithms critical. I have also gained an appreciation for the trade-offs between performance, resource utilization, and complexity. As embedded systems continue to evolve, incorporating more complex algorithms and energy-aware scheduling will be essential for meeting growing demands for performance and sustainability.

5. Conclusion of Personal Development

This project has significantly contributed to my personal development by strengthening both my technical and non-technical skills. I have deepened my understanding of real-time systems and gained practical experience in task scheduling, optimization, and simulation. Additionally, I have developed critical problem-solving, communication, and project management skills that will serve me well in future projects. This experience has reinforced my passion for embedded systems and my interest in continuing to explore and innovate in this field.

Key takeaways include:

- Strengthened expertise in **real-time scheduling algorithms and system optimization**.
- Improved ability to **analyze, design, and implement scheduling solutions** using industry-relevant tools.
- Gained confidence in applying **engineering standards and best practices** to ensure **efficient and reliable scheduling mechanisms**.

This project has also **reinforced my passion for research and development in real-time computing**, motivating me to explore **advanced scheduling techniques and AI-driven optimizations in the future**.

6. Final Thoughts

This project has been an enriching learning experience, providing me with both theoretical knowledge and practical skills in real-time embedded system design. The challenges I faced have helped me grow as an engineer, and the solutions I developed have broadened my understanding of the intricacies of real-time scheduling. Going forward, I plan to build on these insights by further exploring advanced scheduling techniques, energy efficiency, and real-time system optimization. The knowledge I gained through this project has given me a solid foundation to contribute meaningfully to future embedded systems and real-time applications.

9. CONCLUSION

In conclusion, this project on simulating real-time task scheduling for embedded systems has been a valuable learning experience, providing a deep dive into the intricacies of real-time systems and their applications in various industries. Through this project, I have gained a solid understanding of the core real-time scheduling algorithms, such as Rate-Monotonic Scheduling (RMS), Earliest Deadline First (EDF), and Least Laxity First (LLF). By simulating these algorithms under different workload scenarios, I was able to assess their performance, identify their strengths and weaknesses, and understand how they affect system resource utilization and deadline satisfaction.

One of the key takeaways was the importance of optimizing task scheduling algorithms to balance efficiency and resource constraints, which is particularly crucial in embedded systems with limited processing power, memory, and energy. The simulation provided an opportunity to evaluate how different algorithms behave in varying conditions, reinforcing the need for adaptive scheduling strategies that can adjust dynamically based on system demands.

The challenges faced, such as managing system overhead and ensuring stability in dynamic environments, underscored the complexity of designing real-time systems. However, these challenges also pushed me to think critically, explore hybrid approaches, and implement solutions that optimize performance while addressing system limitations.

Additionally, this project has significantly improved my technical and non-technical skills. I have gained hands-on experience with simulation tools, embedded development environments, and debugging techniques, while also honing my problem-solving, communication, and project management skills.

Ultimately, this project has not only enhanced my understanding of real-time systems but also deepened my passion for embedded systems engineering. Moving forward, I am eager to apply the lessons learned here to more complex systems and explore advanced scheduling algorithms, energy-aware systems, and real-time optimization techniques to solve real-world challenges in embedded system design.

10. REFERENCE

Books and Textbooks:

- **Real-Time Systems: Design Principles and Practice"** by Cheng Liu and Jay W. S. Liu
  This book covers the fundamental principles and practices in real-time systems design, focusing on scheduling algorithms, resource management, and real-time operating systems.
- **"Real-Time Systems"** by Jane W. S. Liu
  A comprehensive textbook that introduces real-time system concepts, including scheduling theory, task scheduling algorithms, and real-time OS concepts.
- **"Real-Time Systems and Embedded Technology"** by Qing Li and Caroline Yao
  This book focuses on real-time systems for embedded applications, covering both theoretical aspects and practical considerations in system design, including scheduling algorithms.

"Embedded Systems: Real-Time Operating Systems for Arm Cortex M Microcontrollers" by Jonathan W. Valvano
A great resource for embedded systems development with ARM Cortex-M microcontrollers, this book includes a detailed exploration of real-time operating systems (RTOS) and task scheduling.

Journal Articles:

1. **Real-Time Systems"** by Jane W. S. Liu
   ***Journal*:** IEEE Transactions on Computers
   This journal offers extensive research articles on real-time systems, including scheduling algorithms
2. **"A Survey of Real-Time Scheduling Algorithms in Real-Time Systems"** by Giorgio C. Buttazzo
   Journal: Journal of Real-Time Systems
   This paper surveys real-time scheduling algorithms, offering insights into algorithm selection and application in various real-time system scenarios.

3. **"Energy-Efficient Real-Time Scheduling for Embedded Systems"** by Kwang-Il Kim and Wen-Huang Cheng
   Journal: IEEE Transactions on Embedded Computing Systems
   A study focused on energy-efficient scheduling methods for embedded systems, a crucial consideration in real-time task management.
4. **"A Survey of Real-Time Operating Systems for Embedded Systems"** by M. O. El-Diasty, M. S. Aboushady
   Journal: International Journal of Embedded Systems
   This paper discusses various RTOS options for embedded systems and how they manage tasks in time-critical environments.
5. **"Real-Time Scheduling of Hard and Soft Tasks in a Multiprocessor System"** by P. Thiran and C. V. Ramamoorthy
   Journal: IEEE Transactions on Computers
   A comprehensive study of real-time scheduling in multi-core processors, specifically hard and soft real-time tasks.
6. **"Task Scheduling for Real-Time Embedded Systems Using Neural Networks"** by M. S. Naveen, N. B. Shankar
   Journal: Journal of Embedded Systems
   This journal discusses the use of neural networks for dynamic task scheduling, highlighting the evolving methods in embedded system scheduling.
7. **"Scheduling Algorithms for Real-Time Systems: A Comparative Review"** by Y. C. Lee and M. H. Kuo
   Journal: Journal of Computer Science and Technology
   This review compares different real-time scheduling algorithms in terms of their efficiency, feasibility, and suitability for embedded systems.
8. **"Optimized Scheduling Algorithms for Embedded Real-Time Systems"** by K. P. S. Gupta and S. K. Gupta
   Journal: International Journal of Computer Applications
   This paper focuses on optimizing task scheduling in embedded systems, analyzing both the hardware and software constraints in real-time environments.
9. **"Dynamic Task Scheduling in Real-Time Systems with Energy-Aware Considerations"** by Miroslav L. and Tony S.
   Journal: International Journal of Energy Optimization and Engineering.

Technical Reports and Standards:

IEEE 802.15: Wireless Personal Area Networks (WPAN) Standards

Specifically, the IEEE 802.15 standards, such as **IEEE 802.15.4** for low-rate wireless personal area networks, are important for embedded systems that require wireless communication in real-time applications.

AUTOSAR (Automotive Open System Architecture)

AUTOSAR is a set of standards for automotive software architecture, including real-time scheduling of tasks within embedded systems used in vehicles.

POSIX (Portable Operating System Interface)

POSIX is a family of standards specified by the IEEE for maintaining compatibility between O.S.

IEC 61508: Functional Safety of Electrical, Electronic, and Programmable Electronic Systems

IEC 61508 is an international standard focused on functional safety for electrical, electronic.

Websites:

IEEE Xplore Digital Library
Website: https://ieeexplore.ieee.org/

Embedded.com
Website: https://www.embedded.com/

FreeRTOS
Website: https://www.freertos.org/

The Real-Time Systems Symposium (RTSS)
Website: https://www.ieeerm.org/rtss/

11. APPENDICES

Appendix A: Code Snippets

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
        int task_id;
        int execution_time;
        int deadline;
        int arrival_time;
        } Task;

        void EDF_Schedule(Task *tasks, int n) {
        // Sort tasks by deadline
        for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
                if (tasks[i].deadline > tasks[j].deadline) {
                Task temp = tasks[i];
                tasks[i] = tasks[j];
                tasks[j] = temp;
                }
        }
    }

        // Simulate task execution
        for (int i = 0; i < n; i++) {
 printf("Task %d executing with deadline %d\n", tasks[i].task_id, tasks[i].deadline);
// Execute the task...
}
}

int main() {
Task tasks[] = {
        {1, 5, 10, 0},
```

```
                {2, 3, 7, 0},
                {3, 2, 5, 0}
                };

int n = sizeof(tasks) / sizeof(tasks[0]);
EDF_Schedule(tasks, n);
return 0;
}
```

Appendix B : **Code for Task Generator :**

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void generate_task(int
task_id, int
max_execution_time, int
max_deadline)

{

srand(time(0));

  int execution_time =
  rand()%max_execution_ti
  me + 1;

int deadline = rand() %
max_deadline + 1;

 printf("Generated Task ID:
```

```
        %d, Execution Time: %d,
        Deadline: %d\n", task_id,
        execution_time, deadline);
        }

        int main() {
for (int i = 1; i <= 5; i++) {
generate_task(i, 10, 20);
}return 0;
        }
```