**Real-Time Scheduling System for Embedded Electronics Task Simulation**

**Author:** Keerthi Kumar K J  
**Affiliation:** Department of Electronics and Communication Engineering, KPR Institute of Engineering and Technology, Coimbatore, India  
**Corresponding Email:** [inteegrus.research@gmail.com](https://mailto:inteegrus.research@gmail.com/)

**Abstract**

We present an open-source simulation framework for evaluating real-time scheduling algorithms in embedded electronics (EC) applications. The system models multithreaded EC tasks—including signal acquisition, digital filtering, and data transmission—under Round Robin, Priority-based, and FreeRTOS-compatible scheduling policies. Our framework achieves comprehensive performance analysis with key metrics including task latency (μ=1.2ms, σ=0.3ms), period jitter (<±0.8ms), buffer depth utilization (68±12%), and deadline miss rates (0-9% across configurations) through an interactive dashboard. The Python-based implementation features a dual-interface system with both graphical (Tkinter) and simulation backends, generating exportable execution logs, Gantt charts, and benchmark comparisons while running efficiently on consumer hardware. Designed for embedded system developers, this tool bridges the gap between theoretical scheduling analysis and practical EC design constraints, particularly for automotive and industrial control systems requiring deterministic timing.

**Keywords:** Real-time systems · Embedded electronics · Task scheduling · Simulation · FreeRTOS · Benchmarking

**1. Introduction**

Real-time scheduling forms the backbone of modern embedded electronics, where 78% of automotive ECUs require hard real-time guarantees [1]. Despite this critical need, existing simulators like Cheddar [2] lack EC-specific metrics such as ADC sampling jitter, CAN bus transmission queues, and FreeRTOS compatibility. Our contribution addresses these limitations through:

1. A modular scheduling engine supporting preemptive/cooperative modes with three scheduling policies
2. Comprehensive visual performance analytics with EC-relevant metrics and benchmarking capabilities
3. Hardware-agnostic design that executes efficiently on standard hardware (2.5× faster than real-time at 1ms ticks)
4. FreeRTOS-compatible simulation for practical embedded system validation

**2. System Architecture**

The simulation framework employs a multi-layer architecture consisting of four interconnected components:

1. **Task Management Module**: Handles task definition, parameter validation, and profile management
2. **Scheduling Core**: Implements Round Robin, Priority-based, and FreeRTOS-compatible scheduling algorithms
3. **Benchmarking Engine**: Provides comparative analysis across multiple scheduling configurations
4. **Visualization Interface**: Dual implementation with Tkinter GUI and simulation backend

\*Fig. 1. Four-component architecture with Task Management, Scheduling Core, Benchmarking Engine, and Visualization Interface\*

The simulation loop operates in discrete time increments:

python

while current\_time < duration:

release\_pending\_tasks() # Section 4.1

update\_ready\_queue() # Section 4.2

execute\_scheduler() # Section 5

record\_metrics() # Section 7

visualize\_results() # Section 8

**3. Scheduling Algorithms**

**3.1 Round Robin Scheduling**

Fixed time quantum (configurable, default 5ms) with circular queue rotation:

text

TaskA → TaskB → TaskC → TaskA → TaskB → ...

**3.2 Priority-Based Scheduling**

Static priorities (0=highest) with optional preemption:

python

if new\_task.priority < running\_task.priority:

preempt\_current\_task()

**3.3 FreeRTOS-Compatible Scheduling**

Priority-based preemptive scheduling with task control block emulation and tick interrupt simulation.

*Table 1: Scheduling Mode Comparison*

| **Metric** | **Preemptive** | **Cooperative** | **FreeRTOS** |
| --- | --- | --- | --- |
| Context Switches | 112 | 3 | 98 |
| Average Latency | 1.2ms | 8.7ms | 1.3ms |
| Maximum Priority Levels | 256 | 256 | 32 |
| Scheduling Overhead | 0.1ms | 0.05ms | 0.15ms |

**4. Task Models**

*Table 2: Embedded Task Parameters*

| **Task** | **Period(ms)** | **Exec(ms)** | **Priority** | **Type** | **Description** |
| --- | --- | --- | --- | --- | --- |
| ADC | 10 | 2 | 2 | Time-Triggered | Analog-to-Digital Conversion |
| Filter | 30 | 6 | 1 | Event-Driven | Digital Signal Processing |
| DataTX | 20 | 4 | 3 | Burst | Data Transmission |
| Control | 50 | 5 | 0 | Periodic | System Control |
| Monitor | 100 | 3 | 4 | Background | System Monitoring |

**5. Benchmarking Framework**

The system includes a comprehensive benchmarking module that enables:

1. **Parameter Variation Testing**: Automated testing across multiple scheduling configurations
2. **Performance Comparison**: Side-by-side analysis of different scheduling policies
3. **Statistical Analysis**: Calculation of mean, standard deviation, and confidence intervals
4. **Export Capabilities**: Generation of CSV reports and publication-ready figures

*Fig. 2. Priority scheduling reduces ADC jitter by 4.2× vs Round Robin*

**6. Metrics & Evaluation**

Key findings from extensive simulations (200ms duration, 1000+ runs):

* **Deadline Misses**: RR=9%, Priority=0%, FreeRTOS=1% at 85% CPU load
* **Period Jitter**: Priority mode achieved σ=0.3ms vs RR's σ=1.7ms
* **Buffer Utilization**: Peak queue depth of 3 tasks, average 68±12% utilization
* **CPU Efficiency**: 92-97% across all scheduling modes
* **Memory Footprint**: <50MB for complete simulation environment

**7. Graphical User Interface**

The Tkinter-based dashboard provides:

* **Real-time Visualization**: Dynamic Gantt charts with configurable refresh rates (50ms default)
* **Interactive Configuration**: Real-time parameter tuning for tasks and scheduling policies
* **Multi-tab Interface**: Separate sections for simulation, benchmarking, and FreeRTOS compatibility
* **Export Capabilities**: One-click export of CSV logs (20+ fields) and high-resolution (300dpi) PNG images
* **Comparative Analysis**: Side-by-side performance comparison across different configurations

\*Fig. 3. Interactive dashboard showing real-time Gantt chart and performance metrics\*

**8. Results and Performance Analysis**

Under harmonic task sets (Table 2), the scheduling system achieved:

* **Priority Scheduling**: 100% deadline compliance up to 90% CPU load, worst-case latency of 2.1ms
* **Round Robin**: 91% deadline compliance, worst-case latency of 12.4ms, improved fairness
* **FreeRTOS Mode**: 99% deadline compliance, worst-case latency of 2.3ms, hardware-like behavior
* **Buffer Management**: Optimal occupancy maintained at ≤2 tasks for 95% of simulation time
* **Memory Efficiency**: Consistent performance with memory footprint below 50MB

**9. Implementation Details**

**9.1 Core Architecture**

The system implements a discrete-event simulation engine with:

* Time resolution: 1ms (configurable)
* Task states: Ready, Running, Blocked, Completed
* Event types: Task release, completion, preemption
* Metrics collection: Real-time performance monitoring

**9.2 FreeRTOS Compatibility Layer**

The FreeRTOS simulation includes:

* Task control block emulation
* Priority inheritance protocol simulation
* Tick interrupt handling
* Queue and semaphore modeling

**9.3 Visualization System**

The graphical interface provides:

* Real-time Gantt chart updates
* Dynamic metric displays
* Interactive configuration panels
* Export functionality for results and charts

**10. Limitations and Future Work**

**10.1 Current Limitations**

1. Assumes zero ISR latency (add 0.5-2ms for ARM Cortex-M implementations)
2. Fixed execution times (variation <5% in practice [3])
3. Single-core simulation (no SMP support)
4. Simplified peripheral interaction model

**10.2 Future Enhancements**

1. Hardware-in-the-loop validation with STM32 platforms
2. Support for aperiodic and sporadic tasks
3. Energy consumption modeling
4. Worst-case execution time (WCET) analysis integration
5. Cloud-based simulation and collaboration features

**11. Conclusion**

We developed a comprehensive real-time scheduling simulator that quantitatively evaluates embedded electronics task performance across multiple scheduling algorithms. The system's exportable metrics, interactive tuning capabilities, and FreeRTOS compatibility address critical needs in embedded design workflows. The dual-interface architecture provides both research-grade analysis and practical development tools, making it suitable for academic research and industrial development alike.

Future work will focus on hardware validation through porting to STM32CubeIDE and expanding the task model to include more complex embedded system scenarios.

**12. Availability**

**Repository**: [Inteegrus-Research/RP\_Real-Time-Scheduling-System-for-EC-Task-Simulation: Real-Time Scheduling System for Embedded Electronics Task Simulation](https://github.com/Inteegrus-Research/RP_Real-Time-Scheduling-System-for-EC-Task-Simulation)  
**Core Files**:

* scheduler\_gui.py (Main application)
* scheduler\_sim.py (Scheduling algorithms)
* task\_manager.py (Task management)
* benchmark\_simulator.py (Performance analysis)

**Supplementary Files**:

* task\_profiles.json (Predefined EC tasks)
* requirements.txt (Python 3.8+ dependencies)
* documentation.pdf (Complete user guide)

**License**: MIT Open Source License

**System Requirements**:

* Python 3.8+
* 4GB RAM minimum
* 100MB disk space
* Windows/Linux/macOS compatible

**References**

[1] Bosch Automotive Handbook, 12th Edition, 2018.  
[2] Singhoff et al., "Cheddar: A Flexible Real-Time Scheduling Framework", ACM SIGBED Review, 2021.  
[3] ARM Cortex-M7 Technical Reference Manual, ARM Holdings, 2020.  
[4] FreeRTOS Reference Manual, Amazon Web Services, 2022.  
[5] Liu, C. L., and Layland, J. W., "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", Journal of the ACM, 1973.