FreeRTOS on STM32 for Embedded Real-Time Task Management

Ngọ Quang Phú
Thai Nguyen University of Information and Communication Technology
Thai Nguyen University, Thai Nguyen, Vietnam
student551@ictu.edu.vn

***Abstract*—This undergraduate thesis investigates FreeRTOS task management on an STM32 microcontroller. Embedded applications often need predictable task scheduling and controlled exchange of data between concurrent tasks. The author reviews real-time operating-system concepts and scheduling approaches, then implements STM32 exercises using FreeRTOS tasks, queues, and semaphores. The thesis documents the examples and their intended synchronization behavior. The manuscript reports successful execution of the task, queue, and semaphore exercises. These examples demonstrate that the chosen platform can run the studied functions. The source does not provide a measured worst-case timing analysis or industrial field validation. Functional examples alone do not establish timing guarantees, resource limits, or long-term stability. The author identifies wireless communication and more complex resource management as future extensions. The contribution is a documented student study or prototype whose stated scope can guide further testing.**

***Index Terms*—FreeRTOS, STM32, embedded systems, task scheduling**

I. Introduction

Embedded applications often need predictable task scheduling and controlled exchange of data between concurrent tasks. This thesis investigates FreeRTOS task management on an STM32 microcontroller. It begins from a practical engineering need and explains the technologies and workflow chosen for the case. The objective is to document a student implementation or comparative investigation, together with the evidence available in the manuscript. Broader operational or economic effects require separate measurements and are not assumed from the existence of a prototype [1]. Earlier ICTU student work on related designs is included here for context [2], [3]. Related published studies provide context for the subject and methods [4], [5], [6], [7], [8], [9], [10]. These contextual works were selected for this English summary; they are not asserted to appear in the original thesis bibliography.

II. Methods

The author reviews real-time operating-system concepts and scheduling approaches, then implements STM32 exercises using FreeRTOS tasks, queues, and semaphores. The thesis documents the examples and their intended synchronization behavior. The source is summarized through its stated design, implementation material, examples, and conclusion. Features shown in the manuscript are distinguished from proposals for later development. This review does not add a new experiment, fill missing measurements with estimates, or treat a planned capability as an observed result. The source thesis remains the provenance for the reported work [1].

Supervisor: Ngô Thị Vinh, Thai Nguyen University of Information and Communication Technology (ntvinh@ictu.edu.vn).

III. Results

The manuscript reports successful execution of the task, queue, and semaphore exercises. These examples demonstrate that the chosen platform can run the studied functions. The source does not provide a measured worst-case timing analysis or industrial field validation. The report's screens, configurations, examples, or measurements constitute the evidence for this summary. A capability described only as a design intention is not counted as a validated operational outcome. No additional performance, adoption, financial, or safety figure has been introduced here. The result is interpreted within the scope of the student's documented setup [1].

IV. Discussion

Functional examples alone do not establish timing guarantees, resource limits, or long-term stability. The author identifies wireless communication and more complex resource management as future extensions. A later experiment could measure task latency, memory use, and fault behavior while extending the prototype with communication modules. These limitations affect how the evidence should be read. The current manuscript can support understanding of the design and reported demonstration, while independent tests would be needed for stronger claims. Any later deployment should document its environment and separate observed outcomes from intended benefits [1].

V. Conclusion

The thesis documents FreeRTOS task management on an STM32 microcontroller. The manuscript reports successful execution of the task, queue, and semaphore exercises. These examples demonstrate that the chosen platform can run the studied functions. The source does not provide a measured worst-case timing analysis or industrial field validation. Functional examples alone do not establish timing guarantees, resource limits, or long-term stability. The author identifies wireless communication and more complex resource management as future extensions. A later experiment could measure

task latency, memory use, and fault behavior while extending the prototype with communication modules. Accordingly, the present contribution is a source-grounded account of the student work and its tested or illustrated scope. Claims about broader effectiveness should be assessed in later work using explicit evaluation conditions [1].

References

[1] Ngọ Quang Phú, ”FreeRTOS on STM32 for Embedded Real-Time Task Management,” internal undergraduate thesis (original in Vietnamese), Thai Nguyen University of Information and Communication Technology, 2024.

[2] Nông Đức Toàn, ”Construction of Practical Exercises on Raspberry Pi,” internal undergraduate thesis, Thai Nguyen University of Information and Communication Technology, 2023.

[3] ICTU Student 55, ”Design and Construction of a Special-Purpose Computer Based on ATMEGA16 Microcontroller,” internal undergraduate thesis, Thai Nguyen University of Information and Communication Technology, 2008.

[4] Deepak Ramegowda et al., ”Energy efficient mixed task handling on real-time embedded systems using FreeRTOS,” Journal of Systems Architecture, 2022. doi:10.1016/j.sysarc.2022.102708.

[5] Ji-wen DONG et al., ”Improved task scheduling algorithm for embedded real-time operating system and its application,” Journal of Computer Applications, 2009. doi:10.3724/sp.j.1087.2009.02516.

[6] Gessé Oliveira et al., ”Scheduling and energy savings for small scale embedded FreeRTOS-based real-time systems,” Design Automation for Embedded Systems, 2023. doi:10.1007/s10617-023-09267-7.

[7] Moon-Haeng Cho et al., ”Real-Time Task Scheduling Algorithm using a Multi-Dimensional Methodology for Embedded Real-Time Operating Systems,” The Journal of the Korea Contents Association, 2010. doi:10.5392/jkca.2010.10.1.094.

[8] Muhammad Ali Awan et al., ”Intra-task device scheduling for real-time embedded systems,” Journal of Systems Architecture, 2015. doi:10.1016/j.sysarc.2015.07.001.

[9] C. STANGACIU et al., ”An Analysis of a Hard Real-Time Execution Environment Extension for FreeRTOS,” Advances in Electrical and Computer Engineering, 2015. doi:10.4316/aece.2015.03011.

[10] JE Cooling, ”Task scheduling in hard real-time embedded systems using hardware co-processors,” Microprocessors and Microsystems, 1994. doi:10.1016/0141-9331(94)90035-3.