




# Memory Efficiency on the Raspberry Pi 3b Using the Singleton Design Pattern for Murrotal and Salawat Tarhim Application

Mia Fitriawati<sup>1\*</sup> and Mochamad Fajar Wicaksono<sup>2</sup> 

<sup>1</sup>Information System Department, Universitas Komputer Indonesia, Bandung, Indonesia

<sup>2</sup>Computer Engineering Department, Universitas Komputer Indonesia, Bandung, Indonesia

\*miafitriawati@email.unikom.ac.id

**Abstract.** The Raspberry Pi 3 B is used to play MP3 files as a marker for entering prayer times. Murottal or salawat tarhim files will be activated at certain intervals by adjusting the prayer times. Memory limitations on the Raspberry Pi 3B are at risk of crashing, so that the required files fail to load or cannot be played correctly. Meanwhile, the murottal that is played must continue without starting from the beginning when entering the maghrib, dawn, or before Friday prayers. The singleton design pattern is used in the Python programming language to ensure that the system uses only one instance and does not create another instance if the existing instance can still be accessed. By using this singleton design pattern, the system avoids a spike in memory usage on the Raspberry Pi 3 B, leaving sufficient memory space for the system. The goal of this research is to use the singleton design pattern to provide an effective solution for improving memory efficiency and application stability.

**Keywords:** Raspberry Pi 3B, Design Pattern, Singleton Design Pattern, Memory Efficiency.

## 1 Introduction

In software development, code quality is often neglected, with a focus on functionality. Embedded system is a promising solution for improving system efficiency [1]. Embedded systems are computing system with specific functions designed to be a part of larger mechanical or electrical system [2]. The emergence of embedded system and applications specifically designed for specific tasks, especially in the internet of things (IoT) ecosystem, has driven technological development over the past five decades. Unlike typical general-purpose computers, these systems often operate under the real time constrains and with limited resources. Single board computers (SBCs) have a become one of the drivers of this trend, which have provided mobility, flexibility, high level computing power in small form factor, and low latency [3]. Similarly, the previous explanation that SBCs are one of the drivers that offer mobility, flexibility, high level computing power in a small form factor, and low latency, which can be adopted for this research [4].

© The Author(s) 2025

L. Warlina and S. Luckyardi (eds.), *Proceedings of the 8th International Conference on Informatics, Engineering, Science & Technology (INCITEST 2025)*, Advances in Engineering Research 287,

[https://doi.org/10.2991/978-94-6463-924-7\\_3](https://doi.org/10.2991/978-94-6463-924-7_3)

To support all of the explanation before, it will be Raspberry Pi has emerged as one of the single board computers widely used in various projects covering various topics and research fields, including the Internet of Things (IoT), which has gained significant popularity in recent years in various applications [5-6]. Therefore, in this prayer time applications, featuring sa la wat tarhim (recitation from the Quran) and murottal too, will be used to signal the arrival of the call to prayer. It also aims to encourage worshippers to be more prepared for prayer and provide a pre-prayer rest period. This benefit is felt when congregational prayer attendance at mosques increases. Worshippers can also perform sunnah prayers earlier because they have prepared in advance before the call to prayer.

Before the Raspberry Pi 3b was used, reminders were activated using an application installed on Android TV. However, there was a risk that the reminder would fail to activate if the TV was turned off, as it does not automatically turn on in the event of a power outage. Furthermore, the application could not be automatically reloaded when the TV was first turned on. This requires manual action to unlock the system. The Raspberry Pi 3 B is used because it can wake up immediately after a power outage and offers greater flexibility in further configuration, as the device provides an operating system, allowing developers to create applications tailored to their needs. Furthermore, the Raspberry Pi 3B's low-power embedded processor is one way to reduce power consumption in large clusters, rather than using a standard CPU [7].

The Raspberry Pi 3b has a limited 1GB of memory, requiring efficient use. One solution is to reset (reload) the audio instance once the audio has finished being activated. However, some murottal are activated, and reloading them each time is not possible. Reloading frees up some memory slots and causes the audio to start playing from the beginning. The murottal cannot continue to the next verse because the instance has been freed and reloaded.

Design patterns are standard solutions to common design problems [8]. Design patterns are reusable solutions for recurring software design issues. Although useful for software analysis, detecting design patterns is often challenging, especially in large and complex software systems [9]. Seasoned designers frequently utilize past design best practices, codified in the form of design patterns, to make their designs and the resultant code more elegant, robust, and resilient to change [10]. Popularized by the "Gang of Four" (GoF), these design patterns represent best practices codified by experienced designers to produce more elegant, robust, and flexible software [10]. In other words, design patterns describe how objects relate to each other without being tied to specific models and methods.

One of the design patterns introduced by the "Gang of Four" (GoF) is the creational pattern, which has direct relevance to memory management. Among these patterns, the most fundamental and relevant to resource control issues is the Singleton Design Pattern. This pattern ensures that a class uses only one instance and has global access to that instance. The pattern is beneficial in situations where shared resources, such as database connections or configuration managers, are needed across the application [11]. Global variable is to use to make object accessible, but prevent the creation of multiple objects. That class can ensure that no instances are created (by preventing request to create new objects) and provides a way to access the instance. This research aims to be

improved memory singleton design pattern. The singleton design patterns have been proven to provide an effective solution to improve memory efficiency and stability in audio-based application such as murrotal and shalaawat tarhim by ensuring that only one audio management instance is running.

## **2 Literature Review**

### **2.1 Embedded System**

Embedded system has a limitation in terms of energy consumption, data storage, and hardware limitations. Embedded system in this modern's era have become the backbone of the Internet of Things (IoT) and various another device, operating under a series constraints related to cost, power and computing resources. Single board computers (SBCs) offer basic computing functionality at a cost-effective process, making them a good choice for applications [12]. However, for the applications that the demand real-time performance, such as low latency audio processing, using these platforms present significant system challenges. In learning, predictable performance and efficiency memory management are crucial factors in determining the success or failure of these applications. Real time, networked embedded systems serve a crucial two-way bridge between the physical and information's worlds [13].

### **2.2 Raspberry Pi 3b**

The Raspberry Pi 3b is built on a quadcore ARM Cortex A53 processor, which provides sufficient computing power for a wide range of the tasks [12]. Due to the continued popularity and evolution of the Raspberry Pi models and their continued growth, perhaps in the coming decades, several studies evaluating its performance have shown that the Raspberry Pi 3b is designed to bridge the gap between traditional microcontrollers and modern computing as a representative platform. Featuring a 1.2 GHz quad-core ARM processor and 1 GB of RAM, the Raspberry Pi 3B platform is powerful enough to run all operating systems, such as Linux, and supports complex applications written in object-oriented languages.

The existing literature consistently depicts the Raspberry Pi not as a platform with static performance, but rather as an engineering arena where optimization is essential to achieving desired performance targets. Numerous studies have demonstrated that the performance of the Raspberry Pi 3B can be easily modified through a series of hardware and software interventions.

### **2.3 Design Pattern**

Design patterns in software engineering are fundamental concepts that represent general, proven, and reusable solutions to problems that frequently arise in the context of software design [14]. This concept originated from the work of architect Christopher Alexander. It was later adapted from software engineering by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, known as the "Gang of Four" (GoF). GoF

are aims to provide a common language or standard vocabulary. The language enables developers to communicate more effectively about complex design solutions, improve code readability, and ultimately accelerate the development process.

## 2.4 Singleton Design Pattern

The Singleton Pattern can essentially be described as a class that has only one instance and provides a global access point to that single instance [15]. Among the design patterns, the Singleton Design Pattern holds a unique position due to its simplicity and conceptual clarity. Despite its, the conceptual simplicity is implementing the Singleton Design Pattern in the real world presents challenges when dealing with multi-threaded environments. The Singleton Design Pattern is typically used to ensure that only one instance has global access [16]. Several literature and technical studies have identified several implementation variants, each with its own advantages and disadvantages [17]. The specific features used in the Singleton Design Pattern provide a detailed pattern definition and allow for the identification of non-standard implementations [17].

## 3 Method

This section will explain the methodology that will be used to the research hypothesis. The use of experimental design, as a methodology in this section, is used to ensure that the results are replicable, valid, and directly address the research objectives. An experiment can be said to be fundamental to scientific and engineering practice. A well-designed experiment can produce an empirical model of a process, facilitating the understanding and prediction of its behavior [19]. Therefore, an experimental design can be said to be the backbone of scientific investigations that aim to establish cause and effect relationship.

After completing the first steps outlined above, the next step is to conduct an experimental procedure designed to ensure consistent and controlled testing that minimizes confounding variables and produces the desired data. The test was conducted by monitoring murotal (recitation) activity performed before tarhim for the Fajr and Maghrib prayers. This test was conducted at a mosque in the Kamyangan Residence complex. Tarhim would be played for 10 minutes before the call to prayer. The experimental method involved environmental preparation, consisting of a Raspberry Pi 3B unit with the Murotal and Tarhim applications installed. Using this platform ensures that the results obtained are directly relevant to the topic. From an environmental perspective, the platform will consistently install and utilize cooling to mitigate CPU performance degradation caused by high temperatures, which can impact performance metrics. From an environmental perspective, the results, which occurred over two days, resulted in numerous issues, posing a challenge. Given these challenges, the researchers attempted to use the singleton design pattern to address the issues (see Fig. 1).



**Fig. 1.** The Raspberry Pi 3B Device.

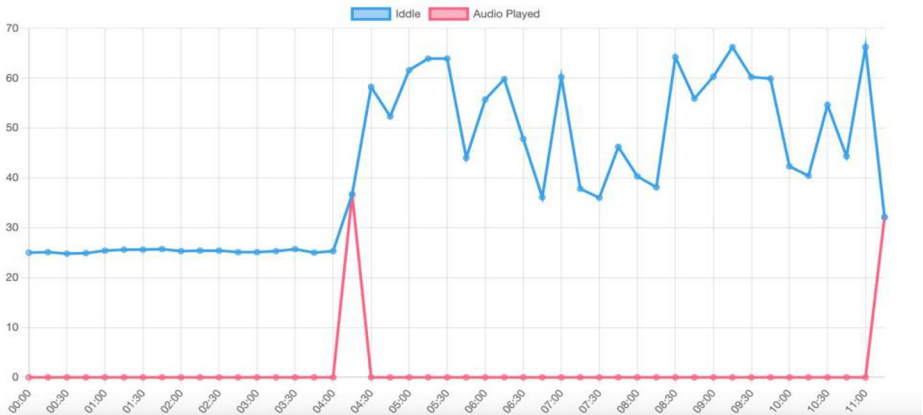
After completing the first steps outlined above, the next step is to conduct an experimental procedure designed to ensure consistent and controlled testing that minimizes confounding variables and produces the desired data. The test is conducted by monitoring activity on murotal (recitations) performed before tarhim for the dawn and evening prayers. The tarhim will be played for 10 minutes before the call to prayer. The experimental method involves preparing the environment, consisting of a Raspberry Pi 3B unit with the Murotal and Tarhim applications installed. Using this platform ensures that the results obtained are directly relevant to the topic. From an environmental perspective, the platform will consistently install and utilize cooling to mitigate CPU performance degradation caused by high temperatures, which can impact performance metrics. Standardizing the environment will be key in configuring the hardware and software for testing.

The test results, in the form of measurement metrics and data analysis, will be explained in the results and discussion section, along with the metric depictions in Figs. 2 and 3. The measurement metrics used reflect the impact of the instantiation strategy on memory usage. The selection of metrics aims to provide comprehensive results regarding memory efficiency, not only in terms of quantity but also in terms of overhead management. Data analysis from the testing and measurement metrics is then compiled to determine whether the results align with the study's objectives.

## 4 Results and Discussion

The design pattern is implemented using Python and installed on a Raspberry Pi 3B device. The Raspberry Pi 3B is installed in conjunction with an audio amplifier as the sound device. The following image illustrates the use of the Raspberry Pi 3B for the Murotal and Tarhim applications. Fig. 2 shows a graph obtained from the system

showing memory usage before using the singleton. The graph shows a fluctuating increase in memory compared to before the audio started playing. Then, a memory drop occurs just before the audio starts playing in the second interval due to the service being terminated. Terminating the service is intended to prevent a system crash due to lack of memory.



**Fig. 2.** Dashboard usage test before used singleton design pattern.

Fig. 3 illustrates the system graph after using the singleton design pattern. Fig. 3 shows that the system is more stable, eliminating the need for stopping and preventing potential crashes. Therefore, using the Singleton Method Design Pattern ensures that a class has only one instance and provides a global access point for it. This pattern is ideal for scenarios that require centralized control, such as managing database connections or configuration settings.

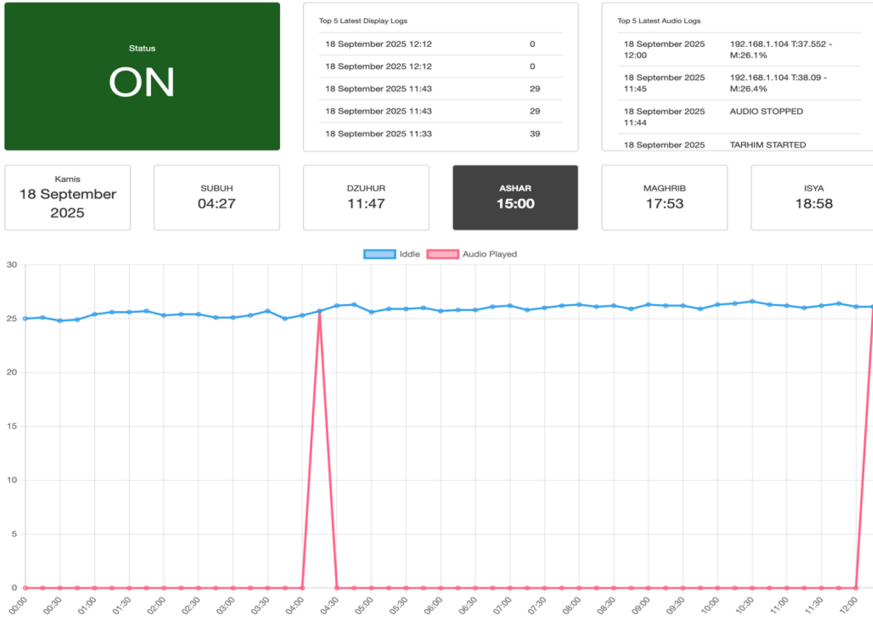


Fig. 3. Dashboard usage test after used singleton design patter.

A web dashboard page was built to monitor system performance. Testing was conducted under three conditions: when the system was running in an idle state, when audio was enabled, and when the system was restarted. Fig. 4 shows the results of the application usage test.



Fig 4. Dashboard usage test.

The graph on the dashboard displays the amount of memory used when the device is turned on, when audio is enabled, and when memory is in use. The blue line indicates when the system was restarted and the amount of memory used by the system. The orange line indicates the amount of memory used when audio was enabled. The red line indicates the amount of memory used when the system is idle. If the red line decreases and meets the blue line, it indicates the amount of memory after the system has been restarted. The intersection with the orange line indicates the amount of memory used when audio is played.

Based on test results, system memory usage, especially when audio is playing, does not show any spikes. Memory can be released when the audio is finished. However, this requires reloading the audio and starting over. The activated file, however, is a murottal (song recitation) that continues and is only paused for a certain period. A similar study conducted by Azizi et al. in 2025 showed that it significantly improves time efficiency without increasing memory load [20]. Thus, it can be concluded that by measuring the increase in application performance, which is indicated by a decrease in RAM usage and better stability, after implementing Singleton for audio management.

## 5 Conclusion

The idea of the design pattern is using existing variable or object instead of creating the new one. Based on the test result, the graph shows that the Singleton Design Pattern for MP3 file playback can also reduce the memory usage. The hardware used to test - Raspberry Pi 3B's – that has 1 GB memory limitation can be utilized efficiently without requiring the need to reload files, pause audio, or restart the system.

## References

1. Ananda, R., Amin, M., Lubis, I.A.: PEMANFAATAN TEKNOLOGI EMBEDDED SYSTEM DIBIDANG PERTANIAN BERBASIS SENSOR. *Jurnal Pemberdayaan Sosial dan Teknologi Masyarakat*. **5**(1), 266 (2025). <https://doi.org/10.54314/jpstm.v5i1.3817>.
2. Beningo, J.: *Embedded Software Design: A Practical Approach to Architecture, Processes, and Coding Techniques*. Apress (2022).
3. Wevolver Robotics, <https://www.wevolver.com/article>, last accessed 2025/1/14
4. Gamess, E., Hernandez, S.: Performance Evaluation of Different Raspberry Pi Models for a Broad Spectrum of Interests. *International Journal of Advanced Computer Science and Applications*. **13**(2), (2022). <https://doi.org/10.14569/ijacsa.2022.0130295>.
5. Ramasamy, L.K., Kadry, S.: *Blockchain in the Industrial Internet of Things*. IOP publishing, India (2021)
6. Lima, D.B.C., da Silva Lima, R.M.B., de Farias Medeiros, D., Pereira, R.I.S., de Souza, C.P., Baiocchi, O.: A Performance Evaluation of Raspberry Pi Zero W Based Gateway Running MQTT Broker for IoT. In: 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). IEEE (2019).
7. Hosny, K.M., Salah, A., Magdi, A.: Parallel Image Processing Applications Using Raspberry Pi. In: *Studies in Computational Intelligence*. pp. 107–119. Springer International Publishing, Cham (2023).

8. Bijlsma, L.A., Kok, A.J.F., Passier, H.J.M., Pootjes, H.J., Stuurman, S.: Evaluation of design pattern alternatives in Java. *Software: Practice and Experience*. 52, 1305–1315 (2021). <https://doi.org/10.1002/spe.3061>.
9. Moreira, R., Fernandes, E., Figueiredo, E.: Based Comparison of Design Pattern Detection Tools. In *29th International Conference on Pattern Languages of Programs (PLOP)*. pp. 1-16. Association for Computing Machinery, Brazil (2022)
10. Qamar, N., Malik, A.A.: Impact of Design Patterns on Software Complexity and Size. *Mehran University Research Journal of Engineering and Technology*. 39(2), 342–352 (2020). <https://doi.org/10.22581/muet1982.2002.10>.
11. Sabbag Filho, N.: Análise Comparativa de Padrões: Distinções e Aplicações dos Padrões Comportamentais, Criacionais e Estruturais. *Leaders Tec*, 1(11), 1-5 (2024)
12. Yamanoor, N.S., Yamanoor, S.: High-quality, low-cost education with the Raspberry Pi. In: *2017 IEEE Global Humanitarian Technology Conference (GHTC)*. pp. 1–5. IEEE (2017).
13. Koulamas, C., Lazarescu, M.T.: Real-Time Embedded Systems: Present and Future. *Electronics*. 7(9), 205 (2018). <https://doi.org/10.3390/electronics7090205>.
14. Afreen, J.: A Survey on Artificial Intelligence Techniques to Prevent Cyber Crime. *International Journal of Advanced Research in Computer Science and Software Engineering*. 8, 80 (2018). <https://doi.org/10.23956/ijarcsse.v8i5.669>.
15. Filho, N.: The Singleton Pattern in Scalability Contexts: Performance Evaluation and Maintenance Impacts of Systems, (2024). <https://zenodo.org/doi/10.5281/zenodo.13719389>.
16. Akhmad Nur Hasim, J., Sa'adah, U., Intan Permata Sari, D., Annisa Damastuti, F., Nur Koirudin, F.: Developing Microframework based on Singleton and Abstract Factory Design Pattern. In: *2022 International Electronics Symposium (IES)*. pp. 676–683. IEEE (2022).
17. Stencel, K., Węgrzynowicz, P.: Implementation Variants of the Singleton Design Pattern. In: *Lecture Notes in Computer Science*. pp. 396–406. Springer Berlin Heidelberg, Berlin, Heidelberg (2008).
18. Nazar, N., Aleti, A., Zheng, Y.: Feature-based software design pattern detection. *Journal of Systems and Software*. 185(1), 111179 (2022). <https://doi.org/10.1016/j.jss.2021.111179>.
19. Greenhill, S., Rana, S., Gupta, S., Vellanki, P., Venkatesh, S.: Bayesian Optimization for Adaptive Experimental Design: A Review. *IEEE Access*. 8(1), 13937–13948 (2020). <https://doi.org/10.1109/access.2020.2966228>.
20. Azizi, B., Pratama, A.A., Dysa, G.M.A., Carudin, C.: Analisis Penerapan Design Pattern Singleton Dalam Pengelolaan Koneksi Database Untuk Efisiensi Memori. *Jurnal Informatika dan Teknik Elektro Terapan*, 13(3), 2127-2136 (2025).

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

