

# Implementation of Service Oriented Architecture for Control Systems: A Hardware Demonstration

Muhammad Ashhab Khan<sup>1</sup>, Chayakorn Netramai<sup>1</sup>, Ole Groß<sup>2</sup>

<sup>1</sup> Department of Electrical and Computer Engineering, Sirindhorn International Thai-German Graduate School of Engineering, King Mongkut's University of Technology North Bangkok  
Bangkok, Thailand

s6409106860066@email.kmutnb.ac.th; chayakorn.n@tggs.kmutnb.ac.th

<sup>2</sup> Chair of Computer Science 11 - Embedded Software (Informatik 11 - Embedded Software, RWTH Aachen University)

RWTH Aachen University, Germany  
gross@embedded.rwth-aachen.de

**Abstract** - This paper presents the implementation and evaluation of a Service-Oriented Architecture (SOA) for control systems on an embedded hardware platform, with a specific case study focusing on the Three Tank Control system. The service for the Three Tank Control System is implemented on a microcontroller. Subsequently, this service is integrated with other services running on a separate computing unit. The purpose of this study is to investigate whether executing SOA on embedded devices with limited resources is feasible. By examining the adaptability and potential benefits of SOA in the context of low-resource microcontrollers, this research extends the applicability of this architectural approach to a wide range of control applications.

**Keywords:** Service-Oriented Architecture (SOA), Automotive Service-Oriented Architecture (ASOA), Service-Oriented Model-Based Control (SOMC), Real Time Publish Subscribe Protocol (RTPS), Cyber-Physical Systems (CPS)

## 1. Introduction

The field of control systems plays a critical role in managing and regulating various industrial processes and technologies. The evolution of industries has demanded more efficient and complex control systems that can adapt to different scenarios. A service-oriented architecture (SOA) for control systems can be implemented to provide interoperability, flexibility, and reconfigurability. This will simplify the wide range of automation systems and allow developers to focus on algorithm development and creative approaches towards intelligent control.

Service-Oriented Architecture has been applied in different fields. It has been used in robotics software development and also for electronic and web-based applications [1]. It is also being used in enterprises to preserve and reuse their legacy systems and create scalable distributed applications [2]. Additionally, Service-Oriented Architecture is being applied to Cyber-Physical Systems (CPS) to address the challenges of limited resources and complex processes. An instance of this can be seen in the application of Service-Oriented Architecture (SOA) in autonomous cars. This study expands on the Automotive Service-Oriented Software Architecture (ASOA) presented in 2020 [3], within the framework of automated vehicles. The design allows for the division of software components into platform-independent services that may be seamlessly integrated during runtime by a central controller.

[4] proposes a Service-Oriented Model-based Control (SOMC) approach to handle the challenges of complex control systems and automation technologies. It is based on the approach to interpret control loop elements as services that can be flexibly integrated at runtime using an orchestrator. The SOMC approach is then applied to a simulated three-tank system to implement a control loop consisting of services and an orchestrator. It is shown that, using their architecture, control loop elements can be dynamically exchanged at runtime to adapt a control loop to different operating points.

Building on this context, this paper presents the implementation of a three-tank control system service on embedded hardware, specifically STM32F7. The choice of this embedded hardware is based on its affordability, widespread availability, and ease of code porting. By implementing the three-tank control system service on the STM32F7, the functionality and performance of the service-oriented architecture can be evaluated in a real-world hardware environment, ensuring that it can compete effectively with traditional control systems that typically run on less expensive, lower-powered devices.

## 1.1 Problem Statement

Although Service-Oriented Architecture (SOA) has been widely used in many applications, there is a lack of research on its implementation on embedded hardware, specifically in the context of real-time control systems. This research study seeks to fill this research void by concentrating on the execution of Service-Oriented Architecture for control systems and showcasing this architecture on embedded hardware. The Three Tank Control System will be used as a demonstrator system for this study, with the STM32F7 microcontroller being the chosen embedded hardware.

## 2. Background

Monolithic architecture treats the software as one tightly coupled entity. This architecture consolidates the user interface, the business layer (or core algorithm), and the data interface into a single location, rather than dispersing them across multiple modules. The architecture faces scalability issues as any modification requires the entire system to be compiled and deployed [5]. As monolithic applications grow, they become complex and difficult to maintain, increasing maintenance times and scalability costs.

To address these challenges, other approaches like Service-Oriented Architecture (SOA) are employed. This includes microservices and serverless computing because they offer improved scalability, maintainability, and resource allocation. Services in SOA are self-contained units of functionality. These services can be distributed throughout the network and controlled by different ownership domains. They interact with users through information exchanges and rely on standard protocols for interoperability.

### 2.1. Automotive Service Oriented Architecture (ASOA)

One example of Service-Oriented Architecture is Automotive Service Oriented Architecture [3]. Automotive Service-Oriented Architecture (ASOA) acts as a solution to the challenges faced in the automotive and networked driving domains. ASOA enables software to be built using a runtime-integrated service-oriented software architecture, which allows for the integration of components at runtime. ASOA is utilized in the UNICARagil project [6] and also provides organizational tools to aid organizations in development. The general architecture of ASOA is shown in Figure 1.

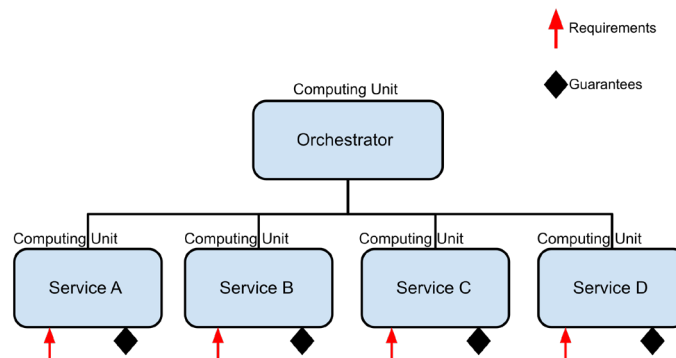


Figure 1: General Architecture of Automotive Service Oriented Architecture (ASOA)

As Figure 1 depicts in ASOA, services can define requirements and guarantees. Requirements specify the inputs that a service requires, while guarantees specify the outputs that a service provides. This ensures that the service inputs and outputs always stay the same, making it easier to integrate services at runtime.

The orchestrator introduced in ASOA is a central controller to achieve runtime-integration of its services. It is responsible for putting the vehicle system into another state. The orchestrator maintains a list of all services and can send commands to these services to start, stop, or connect the services to each other.

### 2.2. Service-Oriented Model-Based Control (SOMC)

The SOMC approach is a method for controlling systems that focuses on providing services and allows for the integration of control loop parts as services in embedded systems. The methodology relies on the Automotive

Service-Oriented Architecture (ASOA). The SOMC approach views control loop parts as services that can be seamlessly incorporated during runtime through an orchestrator. The SOMC orchestrator offers a range of valuable functions for control systems, such as visualization capabilities for services, automated identification of new services, and features for monitoring and managing the system.

An instance where SOA is employed is in the three-tank control system [7]. The three-tank system comprises three interconnected tanks with static orifices, a water supply equipped with an adjustable orifice, and an ultrasonic sensor for measuring the liquid level ( $H_3$ ) in tank three. The authors derive a mathematical model of the three-tank system for linear control theory by approximating the nonlinear differential equations with linear equations.

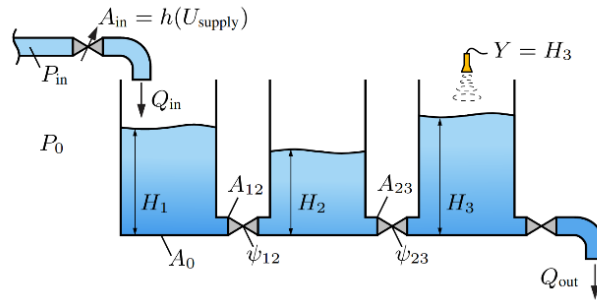


Figure 2: Three Tank Control System (Adapted [7])

[7] applies their Service-Oriented Model-based Control (SOMC) technique to the simulated three-tank system by implementing a control loop that includes services and an orchestrator. The authors establish five functionality types that serve as data representations of the vectors and matrices outlined in the state-space formulation of linear time-invariant (LTI) systems. These functionality types can carry meta information about data quality or service parameters as required. The authors apply their SOMC approach to a simulated three-tank system and establish a control loop comprising services and an orchestrator. For this purpose, six services are defined in the architecture for the three-tank control system, each corresponding to a different control loop component. The connection between these services is shown in Figure 3.

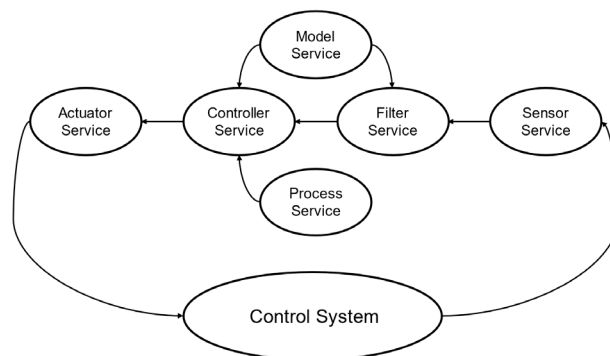


Figure 3: Interconnection of Services for the Three Tank Control System (Adapted [7])

1. The **sensor service** reads data from the sensors used in the three-tank control system.
2. The **actuator service** sends control commands to a physical system.
3. The **controller service** calculates control commands based on measurement data and a control algorithm.
4. The **filter service** filters measurement data to reduce noise or perform other signal processing tasks.
5. The **model service** provides mathematical parameter values to the controller service.
6. The **process service** provides reference values to the controller service.

Furthermore, the SOMC orchestrator has the responsibility of dynamically integrating the various services depicted in Figure 3 during runtime in order to accomplish the intended control loop. The system takes the control loop

configuration as input and utilizes it to create the necessary services. The orchestrator is responsible for overseeing the communication between the services and ensuring the accurate transmission of data between them.

### 3. Implementation

The current implementation of three tank control system is on a Linux machine. The various services of Three Tank Control System as described in section 2.2 are controlled by a Service Oriented Model-Based Control (SOMC) orchestrator.

The SOMC Library is a software library built on top of the Automotive Service Oriented Architecture (ASOA) that is designed to provide a set of common components for all Service-Oriented Model-Based Control (SOMC) projects. The library is intended to make the development of control systems easier by providing a toolbox of functionalities, services, and utility functions that can be used to create a wide range of control systems. The library also aims to implement analysis and evaluation functions that can be used to evaluate every control system that uses this library.

To ensure seamless communication between services, ASOA employs the Real-Time Publish Subscribe Protocol (RTPS) as its middleware. RTPS facilitates efficient data exchange and coordination among the distributed services, enabling the realization of a scalable and interoperable control system architecture.

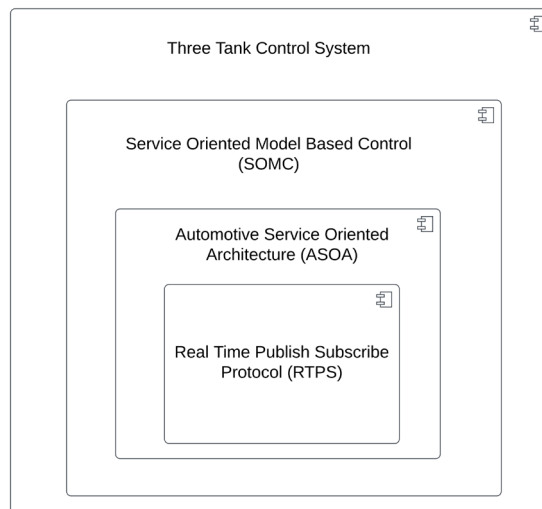


Figure 4: Dependencies of the Three Tank Control Project

#### 3.1. Implementing Three Tank Control System Service on STM32

The three-tank control system, as described in section 2.2, was initially simulated on a Linux machine. Figure 5 illustrates the interaction between the services in the simulated three-tank control system.

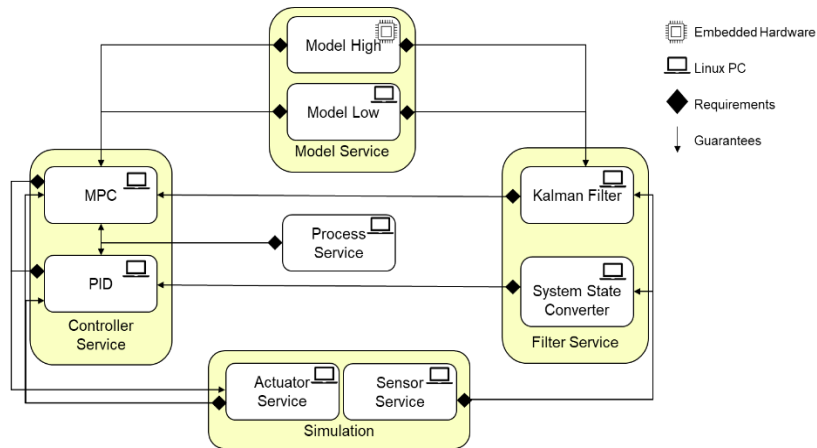


Figure 5: Services and Orchestration of the Simulated Three Tank (Adapted [7])

In Figure 5, the diamond shape represents the requirements (input) of the system, while the arrow depicts the guarantees provided by the services. Specifically, the Model Service guarantees the model parameters for both the Filter Service and the Controller Service. Additionally, the Model High and Model Low services provide parameters for two different operating points of the system. The requirements and guarantees for all the services of the three-tank control system are defined in the same template as those for the ASOA services.

Until this point, the all the services within the three-tank control system has been developed and operated exclusively on a Linux PC. This research paper's primary focus is to detail the implementation of the "Model High" service on the STM32F7 microcontroller. Subsequently, this STM32-based service will establish communication with the remaining services hosted on the Linux machine.

### 3.2. Implementation Methodology

The implementation of SOA on STM32 consists of multiple interconnected steps. The STM32F767 microcontroller was initially employed to implement embeddedRTPS for the purpose of establishing communication between the services themselves and with the orchestrator. Following that, the Automotive Service Oriented Architecture (ASOA) was implemented on the STM32F7, enabling the creation and exchange of services. Subsequently, the Service Oriented Model Based Control (SOMC) mechanism was implemented on ASOA to regulate the operations of the three-tank control system. After this, the "Model High" service of the three-tank control system was implemented. Finally, the service was successfully combined with other services operating on the other processing units. These steps are explained in the following sections.

### 3.3. Running the Real Time Publish Subscribe Protocol on STM32

embeddedRTPS is a portable and open-source C++ implementation of the Real-Time Publish-Subscribe Protocol (RTPS) for embedded systems. The embeddedRTPS is characterized by its portability, primarily owing to its use of lightweightIP and FreeRTOS APIs.

The embeddedRTPS-STM32 repository on GitHub [8] was used to start embeddedRTPS implementation on the STM32F767ZI microcontroller. This repository provided a minimal working example that facilitated communication between the STM32F767ZI and a Linux system employing FastRTPS. The code directory within the repository contained the STM32CubeIDE project, leveraging EmbeddedRTPS and specifically designed for deployment on the STM32Nucleo-F767ZI development board.

#### 2.6.1 ASOA Core and Platform Driver

The ASOA core code contains logging features for debugging purposes. It supports ARM R5, Linux and some other hardware. The code was adjusted to work with STM32F7 processors as well. This way, all the important information can be sent to the Universal Synchronous/Asynchronous Receiver Transmitter (USART) of the STM32F7.

The implementation of ASOA for ARM R5 uses FreeRTOS as the operating system. Regarding STM32F767, FreeRTOS is also used as the operating system for ASOA. However, due to the hardware differences, specific modifications were required in the ASOA Platform Driver's driver file to ensure compatibility with the STM32F7 platform. These modifications include adapting memory allocation functions, introducing a locking mechanism tailored to FreeRTOS on STM32, and adjusting thread initializations and relevant parameters accordingly.

### 3.4. RTPS Parameters

The parameters for the RTPS were tuned carefully by taking into consideration the memory limitations of the board so that optimal performance and compatibility could be achieved. An essential parameter that needed adjustment was the number of publishers and subscribers that the ASOA framework could accommodate. Since the STM32F767 has limited memory and processing capabilities, it was necessary to determine the appropriate number of publishers and subscribers that the microcontroller could handle without compromising system stability and responsiveness.

Other parameters, such as memory utilization and resource allocation, were also considered during the tuning process. The goal was to strike a balance between the functionality and capabilities of the ASOA framework and the available resources on the STM32F767 microcontroller.

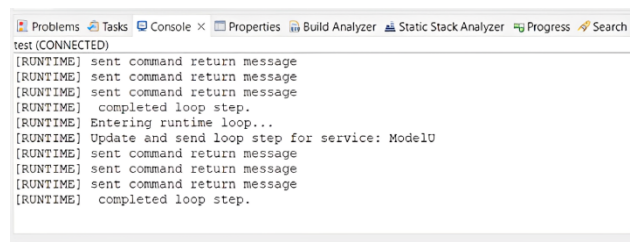
### 3.5. Integrating Services

As the ASOA is implemented on STM32, the next step is to implement SOMC. The SOMC library is developed in C++ with wrappers on top of ASOA and is imported into the STM32 project. The STM32 Nucleo board is connected to a Linux machine via an Ethernet cable for ASOA. The Model High service runs on the STM32, while other services run on the Linux machine. The SOMC Orchestrator facilitates data and control signal exchange among services.

## 3. Results

The implementation of the service on the board produces some useful results. These results are obtained by assessing the functionality evaluation and performance of the system.

The functionality of the service running on the STM32F7 microcontroller was assessed by transferring all the data related to the Automotive Service-Oriented Architecture (ASOA) via the Universal Synchronous/Asynchronous Receiver Transmitter (USART) of the STM32F7. Figure 6 depicts the data received on the USART of the STM32F7. The USART displayed the different procedures that the ASOA follows.



```
Problems Tasks Console x Properties Build Analyzer Static Stack Analyzer Progress Search
test (CONNECTED)
[RUNTIME] sent command return message
[RUNTIME] sent command return message
[RUNTIME] sent command return message
[RUNTIME] completed loop step.
[RUNTIME] Entering runtime loop...
[RUNTIME] Update and send loop step for service: ModelU
[RUNTIME] sent command return message
[RUNTIME] sent command return message
[RUNTIME] sent command return message
[RUNTIME] completed loop step.
```

Figure 6: Data received on USART on STM32

The performance evaluation involves assessing the different resources of the microcontroller during implementation, including memory resources such as RAM and Flash memory. Out of the total 512 kB of RAM available on the STM32F7, a significant portion, precisely 450 kB, was designated for the heap of the FreeRTOS. The allocation of this heap space is critical for managing the execution flow and thread management within the system. Instead of allocating the entire 512 kB, a fraction was set aside as stack memory. The stack is responsible for managing function calls, local variables, and the order in which instructions are executed. On the other hand, the heap enables the allocation of memory that can be changed during the execution of a program. Upon conducting a

thorough examination utilizing STMCubeIDE, it was determined that the overall RAM usage for the entire system, encompassing the SOA and other elements, was 354 kB.

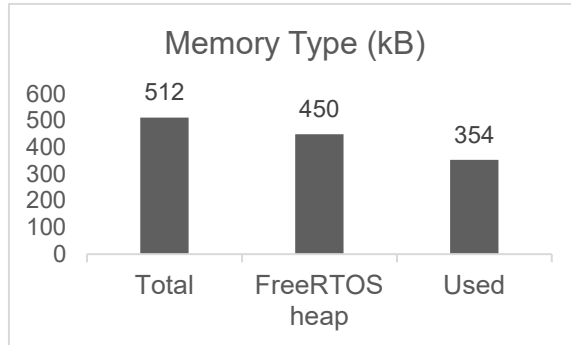


Figure 7: Graph showing RAM usage.

Additionally, an analysis was conducted on the utilization of Flash memory. The STM32F7 has a Flash memory with a total capacity of 2000 kB. During the execution of the SOA, it was determined that 374 kB of the Flash memory was used. The code and data related to the Automotive Service-Oriented Architecture (SOA) and its components did not completely utilize the available Flash memory. Unutilized portions might be reserved for potential future updates, modifications, or additions to the system's software.

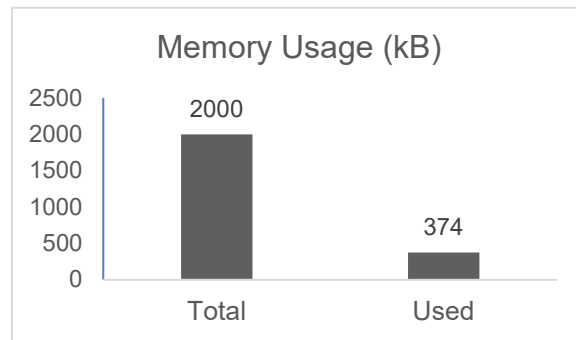


Figure 8: Graph showing flash memory usage.

#### 4. Discussion

The development of a complex service-oriented architecture may be significantly impacted by the STM32F7's limited memory capacity. The system's memory is used up more when there are more services. A single service nearly used up 354 kB of memory, as can be seen from the data. Testing in real-world scenarios has shown that the STM32F7 can support up to two services from the Three Tank Control System before memory limits are surpassed, even though memory usage doesn't grow linearly with the number of services.

The available RAM has a direct impact on how embeddedRTPS parameters are tuned. When developing additional services on the STM32F7, these parameter values might need to be adjusted. In contrast to ARM R5, which benefits from a large memory, while working with a limited memory of 2000 kB, careful attention to the parameter tweaking is necessary.

Integrating external RAM with STM32F7 via the Flexible Memory Controller (FMC) interface that STM32 provides could be one possible solution. SRAM, NOR Flash, NAND Flash, and SDRAM are just a few of the external devices with static memory mapping that can be seamlessly connected with this interface. The requirement for constant parameter tuning may be eliminated by increasing the memory capacity with extra RAM that will allow the maximum value for each parameter.

## 5. Conclusion

In conclusion, this research successfully realized the implementation and demonstration of a Service-Model-Based Control (SOMC) approach for a three-tank control system on the STM32F7 board. The SOMC Orchestrator coordinated services across diverse computing units, enabling seamless interaction between the High" service on STM32F7 and the other services on the Linux machine. However, the limited memory availability on the STM32F7 board posed significant challenges during the porting process, necessitating considerable effort to ensure compatibility.

## References

- [1] Y. Chen and X. Bai, "On Robotics Applications in Service-Oriented Architecture," in *2008 The 28th International Conference on Distributed Computing Systems Workshops*, 2008.
- [2] S. Chandra, "Service oriented architecture for enterprise applications," in *World Scientific and Engineering Academy and Society (WSEAS)*, 2006.
- [3] L. Pischinger and S. Pischinger, "29. Aachen Colloquium Sustainable Mobility : October 5th-7th, 2020, digital event, 1-2; 1. Edition," in *29. Aachen Colloquium Sustainable Mobility = 29. Aachener Kolloquium Fahrzeug- und Motorentechnik*, Aachen, 2020.
- [4] B. Alrifaae and L. Dörschel, "SOMC: Service-Oriented Modell-based Control – Dynamic Software for Dynamic Systems," Deutsche Forschungsgemeinschaft (DFG) - Project number 468483200, 2021.
- [5] S. Ayvaz and Y. B. Salman, "Software Architecture Patterns in Big Data: Transition From Monolithic Architecture to Microservices," in *Applications and Approaches to Object-Oriented Software Design: Emerging Research and Opportunities*, IGI Global, 2020, pp. 90-110.
- [6] A. Kampmann, B. Alrifaae, T. Stolte, I. Jatzkowski, M. Möstl, S. Ackermann, C. Amersbach, D. Püllen, S. Leinen, F. Diermeyer, D. Keilhoff and M. Buchholz, "UNICARagil - Disruptive Modular Architectures for Agile, Automated Vehicle Concepts," *27th Aachen Colloquium*, Aachen, 2018.
- [7] O. Greß, M. Zimmer, D. Scheurenberg, L. Dörschel and B. Alrifaae, "Service-Oriented Model-based Control - Dynamic Software for Dynamic Systems.," 2023.
- [8] embeddedRTPS on STM32Nucleo-F767ZI, Aachen: i11, RWTH Aachen, 2023.