Design Document

Small Operating System (SOS)

Khaled Mustafa Anwar

Sprints

Table of Contents

Introduction

The small operating system brings together a priority-based preemptive scheduler with time-triggered capabilities, designed to optimize task management and system efficiency.

The key feature of our operating system is its priority-based preemptive scheduler, which determines the execution order of tasks based on their assigned priorities. The scheduler constantly monitors the system, interrupting lower-priority tasks when higher-priority tasks need to be executed. This preemptive behavior ensures that critical tasks are promptly attended to, enhancing system responsiveness and meeting time-sensitive requirements.

High Level Design

Layered Architecture

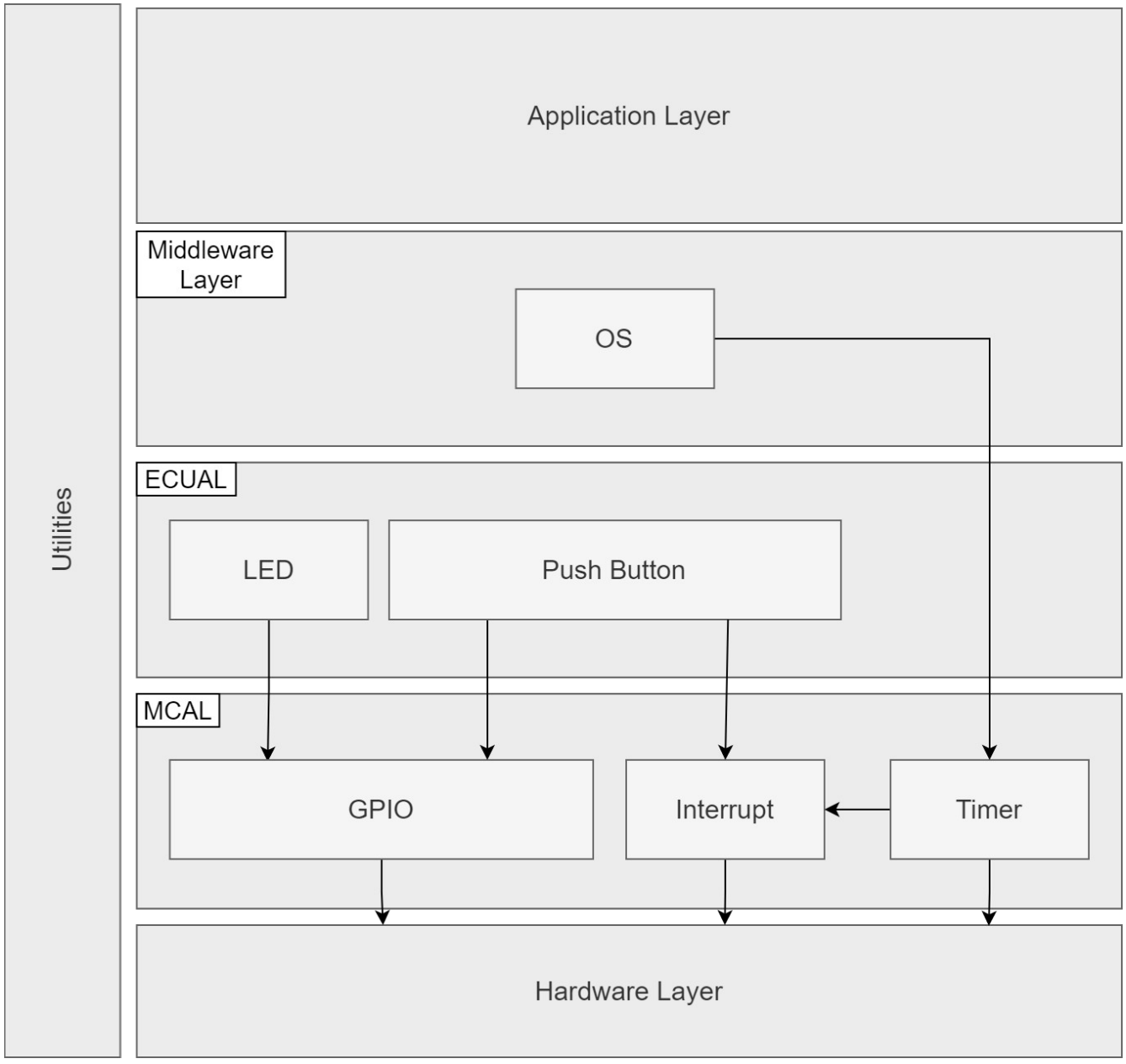

Figure 1: Small OS Scheduler Layered Architecture

Modules’ Description

Application Layer

It defines the functionality and behavior of the system. It is the highest layer in the software stack and encompasses the applications and software components that directly interact with the end-users or external systems.

The primary objective of the application layer is to implement the specific features and tasks that fulfill the intended purpose of the embedded system.

Middleware Layer

The middleware layer serves as a bridge between the application layer and the underlying hardware.

It provides a standardized interface and set of APIs that enable applications to access and utilize hardware resources, such as sensors, actuators, communication interfaces, and memory, without needing to have detailed knowledge of the specific hardware implementation.

Electronic Components Unit Abstraction Layer (ECUAL)

The Electronic Components Unit Abstraction Layer in an embedded systems architecture serves as a bridge between the hardware components and the higher-level software layers. It provides a standardized interface and a set of functions that abstract of individual electronic components, such as sensors, actuators, and peripheral devices.

Microcontroller Abstraction Layer (MCAL)

The Microcontroller Abstraction Layer (MCAL) serves as a bridge between the hardware-specific features of a microcontroller and the higher-level software layers. It provides a standardized set of functions and interfaces that abstract the low-level details of the microcontroller, including its peripherals, timers, interrupts, and input/output (I/O) operations.

Drivers’ Documentation

Application Layer

App_init

Function Name	`App_init`
Syntax	`void App_init(void);`
Parameters[in]	`None`
Parameters[out]	`None`

Parameters[in/out]	None
Return	None

App_start

Function Name	App_start
Syntax	void App_start(void);
Parameters[in]	None
Parameters[out]	None
Parameters[in/out]	None
Return	None

Simple OS (SOS)

SOS_init

Function Name	SOS_init
Syntax	en_system_status_t SOS_init(void);
Parameters[in]	None
Parameters[out]	None
Parameters[in/out]	None
Return	• SOS_STATUS_SUCCESS • SOS_STATUS_INVALID_STATE

SOS_deinit

Function Name	SOS_deinit
Syntax	en_system_status_t SOS_deinit(void);
Parameters[in]	None
Parameters[out]	None
Parameters[in/out]	None
Return	• SOS_STATUS_SUCCESS • SOS_STATUS_INVALID_STATE

SOS_createTask

Function Name	SOS_createTask
Syntax	void SOS_createTask(ptr_sos_db_t *ptr_sos_db, uint8 pid, uint8 priority_lvl);
Parameters[in]	• pid: Process/Task UID • priority_lvl: Priority level of the task

Parameters[out]	ptr_sos_db: Address of the SOS database
Parameters[in/out]	None
Return	None

SOS_deleteTask

Function Name	SOS_deleteTask
Syntax	void SOS_deleteTask(ptr_sos_db_t *ptr_sos_db, uint8 pid);
Parameters[in]	pid: Process/Task UID
Parameters[out]	ptr_sos_db: Address of the SOS database
Parameters[in/out]	None
Return	None

SOS_modifyTask

Function Name	SOS_modifyTask
Syntax	void SOS_modifyTask(ptr_sos_db_t *ptr_sos_db, uint8 pid,uint8 priority_lvl);
Parameters[in]	pid: Process/Task UID
Parameters[out]	ptr_sos_db: Address of the SOS database
Parameters[in/out]	None
Return	None

SOS_run

Function Name	SOS_run
Syntax	void SOS_run(ptr_sos_db_t *ptr_sos_db);
Parameters[in]	ptr_sos_db: Address of the SOS database
Parameters[out]	None
Parameters[in/out]	None
Return	None

SOS_disable

Function Name	SOS_disable
Syntax	void SOS_disable ptr_sos_db_t *ptr_sos_db);
Parameters[in]	ptr_sos_db: Address of the SOS database
Parameters[out]	None
Parameters[in/out]	None
Return	None

LED

LED_init

Function Name	LED_init
Syntax	en_LED_State LED_init(st_LED_config_t *ptr_LED_config);
Parameters[in]	ptr_LED_config: Address to the LED configuration
Parameters[out]	None
Parameters[in/out]	None
Return	• LED_STATUS_SUCCESS • LED_STATUS_FAILED

LED_on

Function Name	LED_on
Syntax	en_LED_State LED_on(st_LED_config_t *ptr_LED_config);
Parameters[in]	ptr_LED_config: Address to the LED configuration
Parameters[out]	None
Parameters[in/out]	None
Return	• LED_STATUS_SUCCESS • LED_STATUS_FAILED

LED_off

Function Name	LED_off
Syntax	en_LED_State LED_off(st_LED_config_t *ptr_LED_config);
Parameters[in]	ptr_LED_config: Address to the LED configuration
Parameters[out]	None
Parameters[in/out]	None
Return	• LED_STATUS_SUCCESS • LED_STATUS_FAILED

LED_toggle

Function Name	LED_init
Syntax	en_LED_State LED_toggle(st_LED_config_t *ptr_LED_config);
Parameters[in]	ptr_LED_config: Address to the LED configuration

Parameters[out]	None
Parameters[in/out]	• LED_STATUS_SUCCESS • LED_STATUS_FAILED
Return	None

Push Button

PB_init

Function Name	PB_init
Syntax	en_PB_State PB_init(st_PB_config_t *ptr_st_PB_config);
Parameters[in]	ptr_st_PB_config: Address of the push button configuration
Parameters[out]	None
Parameters[in/out]	None
Return	PB_STATUS_SUCCESS PB_STATUS_FAILED

PB_status

Function Name	SOS_disable
Syntax	void SOS_disable ptr_sos_db_t *ptr_sos_db);
Parameters[in]	ptr_sos_db: Address of the SOS database
Parameters[out]	None
Parameters[in/out]	None
Return	None

GPIO

GPIO_setPinDirection

Function Name	GPIO_setPinDirection
Syntax	en_GPIO_State GPIO_setPinDirection(st_GPIO_CONFIG_t *ptr_st_GPIO_CONFIG);
Parameters[in]	ptr_st_GPIO_CONFIG: Address of the GPIO pin configuration.
Parameters[out]	None
Parameters[in/out]	None
Return	• GPIO_STATUS_SUCCESS

	• GPIO_STATUS_FAILED

GPIO_writePin

Function Name	GPIO_writePin
Syntax	en_GPIO_State GPIO_writePin(st_GPIO_CONFIG_t *ptr_st_GPIO_CONFIG);
Parameters[in]	ptr_st_GPIO_CONFIG: Address of the GPIO pin configuration.
Parameters[out]	None
Parameters[in/out]	None
Return	• GPIO_STATUS_SUCCESS • GPIO_STATUS_FAILED

GPIO_readPin

Function Name	GPIO_readPin
Syntax	en_GPIO_State GPIO_readPin(st_GPIO_CONFIG_t *ptr_st_GPIO_CONFIG);
Parameters[in]	ptr_st_GPIO_CONFIG: Address of the GPIO pin configuration.
Parameters[out]	None
Parameters[in/out]	None
Return	• GPIO_STATUS_SUCCESS • GPIO_STATUS_FAILED

GPIO_togglePin

Function Name	GPIO_togglePin
Syntax	en_GPIO_State GPIO_togglePin(st_GPIO_CONFIG_t *ptr_st_GPIO_CONFIG);
Parameters[in]	ptr_st_GPIO_CONFIG: Address of the GPIO pin configuration.
Parameters[out]	None
Parameters[in/out]	None
Return	• GPIO_STATUS_SUCCESS • GPIO_STATUS_FAILED

Timer

TIMER_init

Function Name	TIMER_init
Syntax	void TIMER_init(ptr_st_TIMER_CONFIG_t *ptr_st_timer_config)
Parameters[in]	ptr_st_timer_config: Address of the configuration structure of the timer module.
Parameters[out]	None
Parameters[in/out]	None
Return	None

TIMER_start

Function Name	TIMER_start
Syntax	void TIMER_start(en_TIMER_ID_t timer_id)
Parameters[in]	timer_id: Timer ID
Parameters[out]	None
Parameters[in/out]	None
Return	None

TIMER_stop

Function Name	TIMER_stop
Syntax	void TIMER_stop(en_TIMER_ID_t timer_id)
Parameters[in]	timer_id: Timer ID
Parameters[out]	None
Parameters[in/out]	None
Return	None

TIMER_delay

Function Name	TIMER_delay
Syntax	void TIMER_delay(uint32 delay_ms)
Parameters[in]	delay_ms: Specified time for delay in milli-seconds
Parameters[out]	None
Parameters[in/out]	None

Return	None

UML System Diagrams

Class Diagram of the System

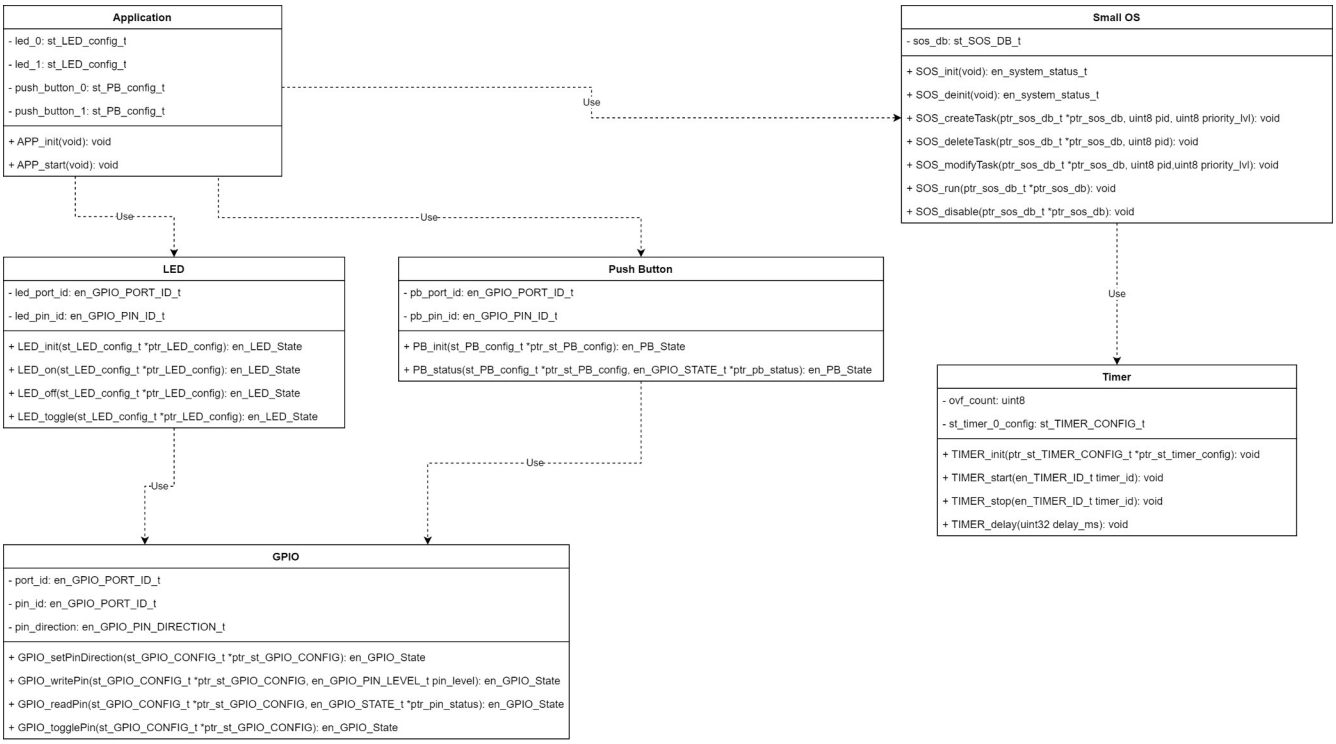

Figure 2: Small OS Class Diagram

State Machine Diagram

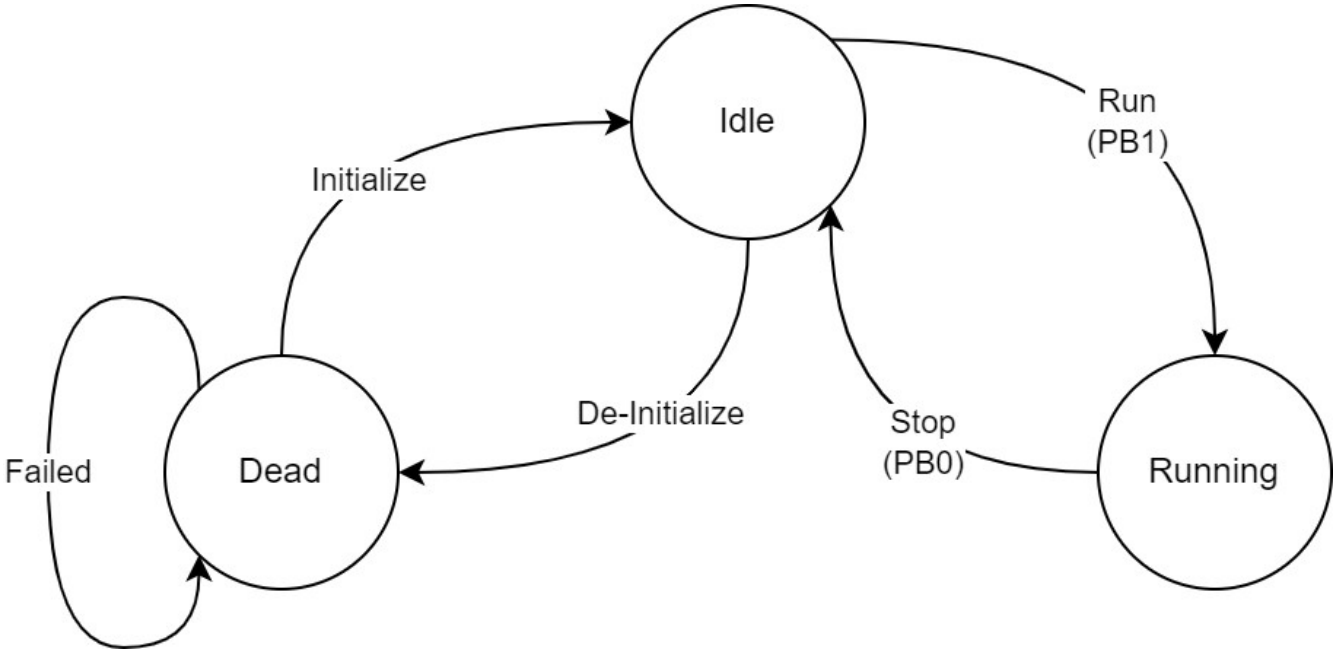

Figure 3: Small OS Scheduler Class Diagram

Sequence Diagram

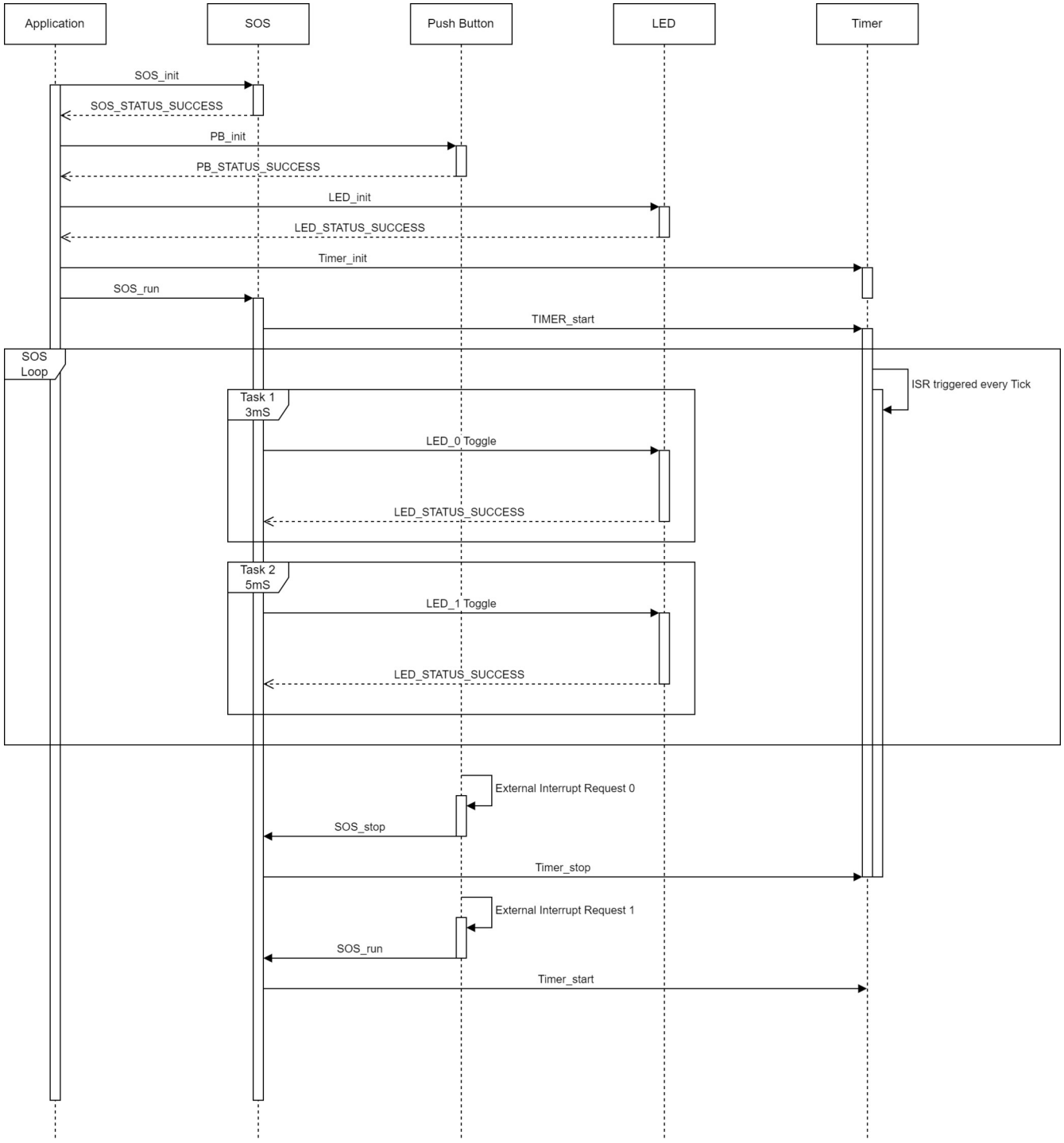

Figure 4: Small OS Scheduler Sequence Diagram