SMALL OPERATING SYSTEM DESIGN

Presented to: Sprints
Prepared by: Ahmed Hesham

Contents

SMALL OPERATING SYSTEM

1. Project Introduction

This project is aiming to deliver a SOS -Small Operating System- which will manage the scheduling of some tasks. The project will be resembling RTOS and for the delivery it will tested on some output/input modules which will toggle LEDs at different periodicities and will be checking on the buttons' states too.

1.1. Project Components

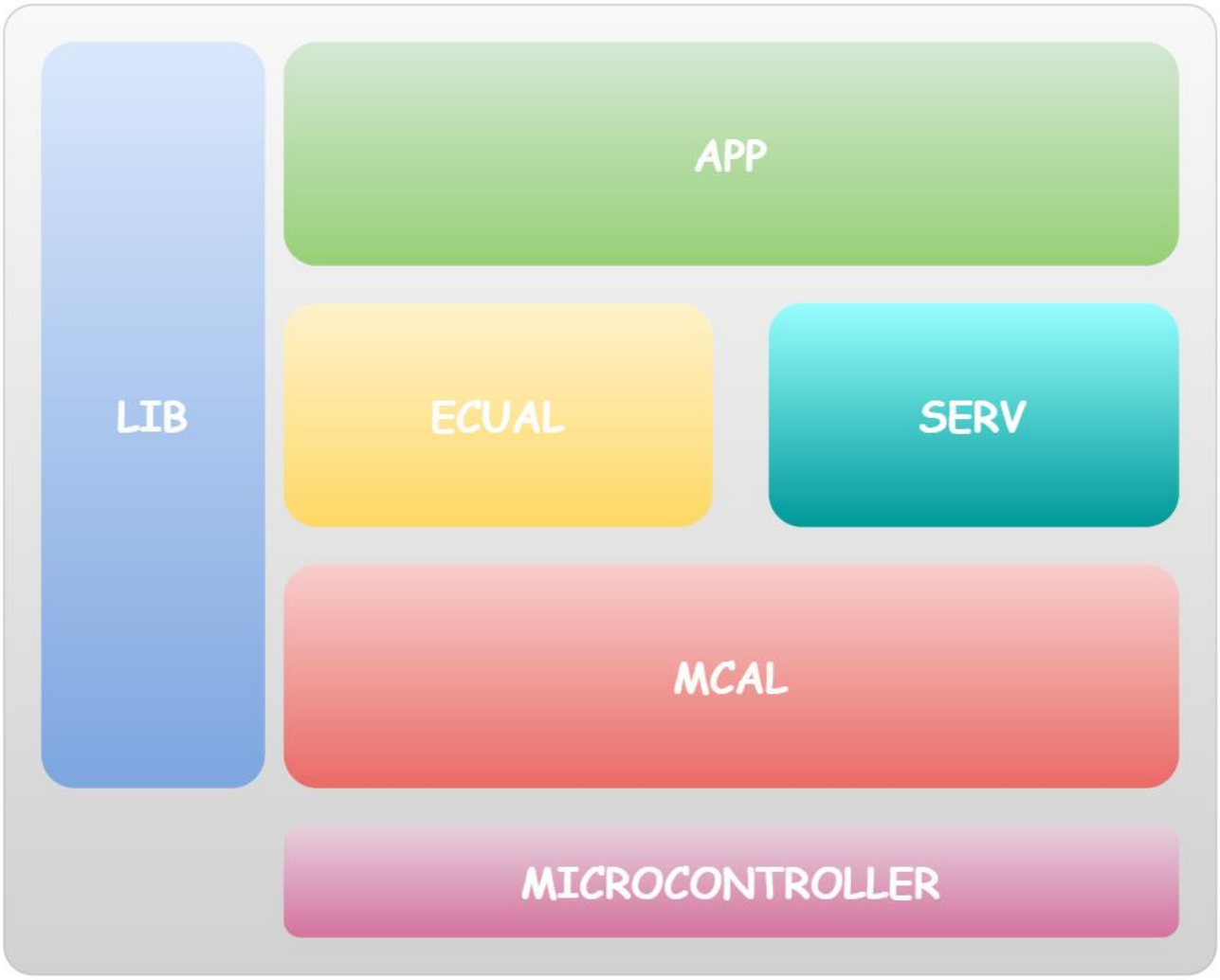
- ATmega32 microcontroller
- 2 LEDS
- 2 Buttons

So I will be using ATmega32 microcontroller for generating timer interrupts which will be used as system tick, will create two different tasks for the LEDs, 1 for STOP button, and START button will be checking on it in the super loop of the application.

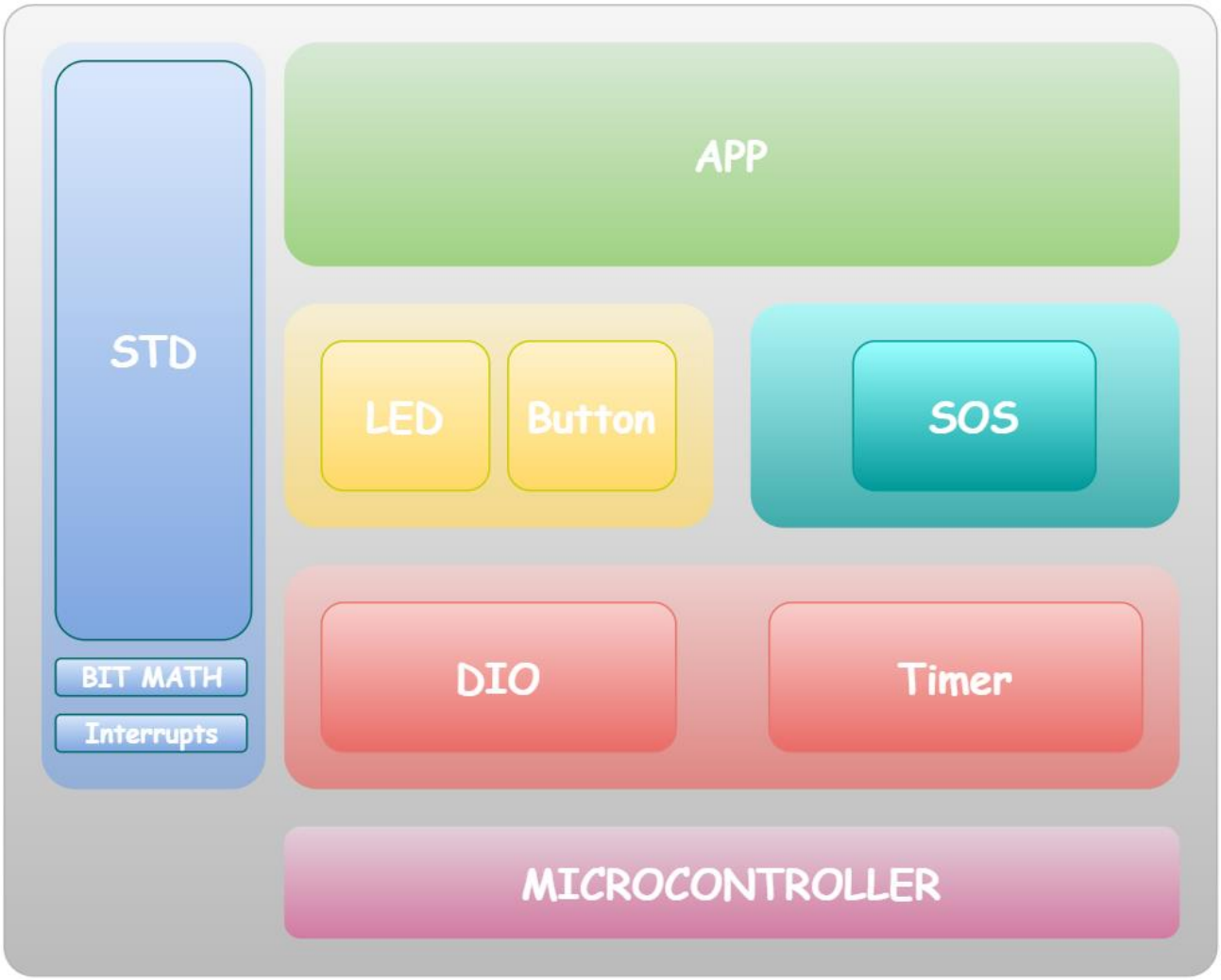
2. High Level Design

2.1. System Architecture

2.1.1. Layered Architecture



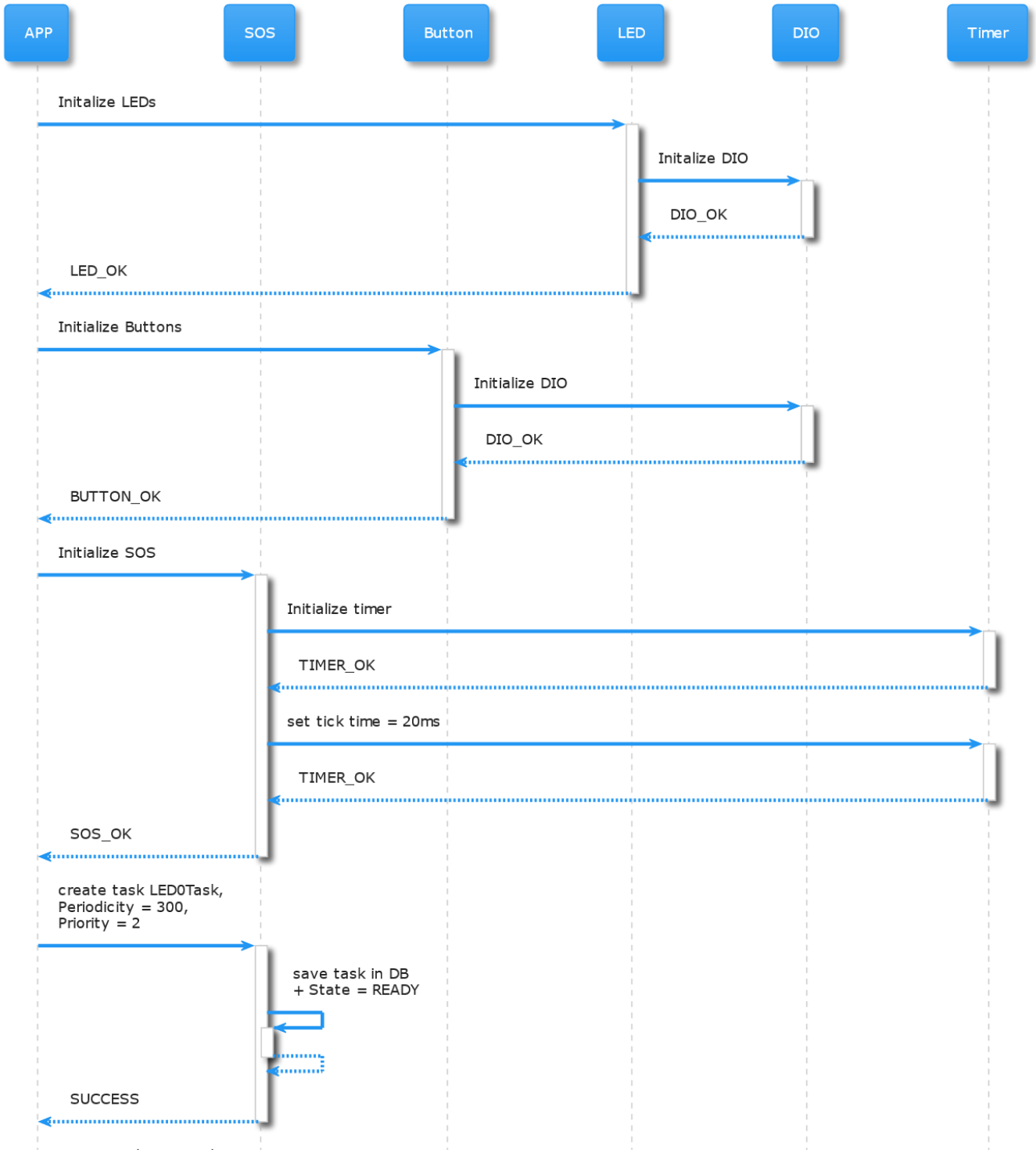
2.1.2. Design


APP
STD
LED
Button
SOS
BIT MATH
Interrupts
DIO
Timer
MICROCONTROLLER

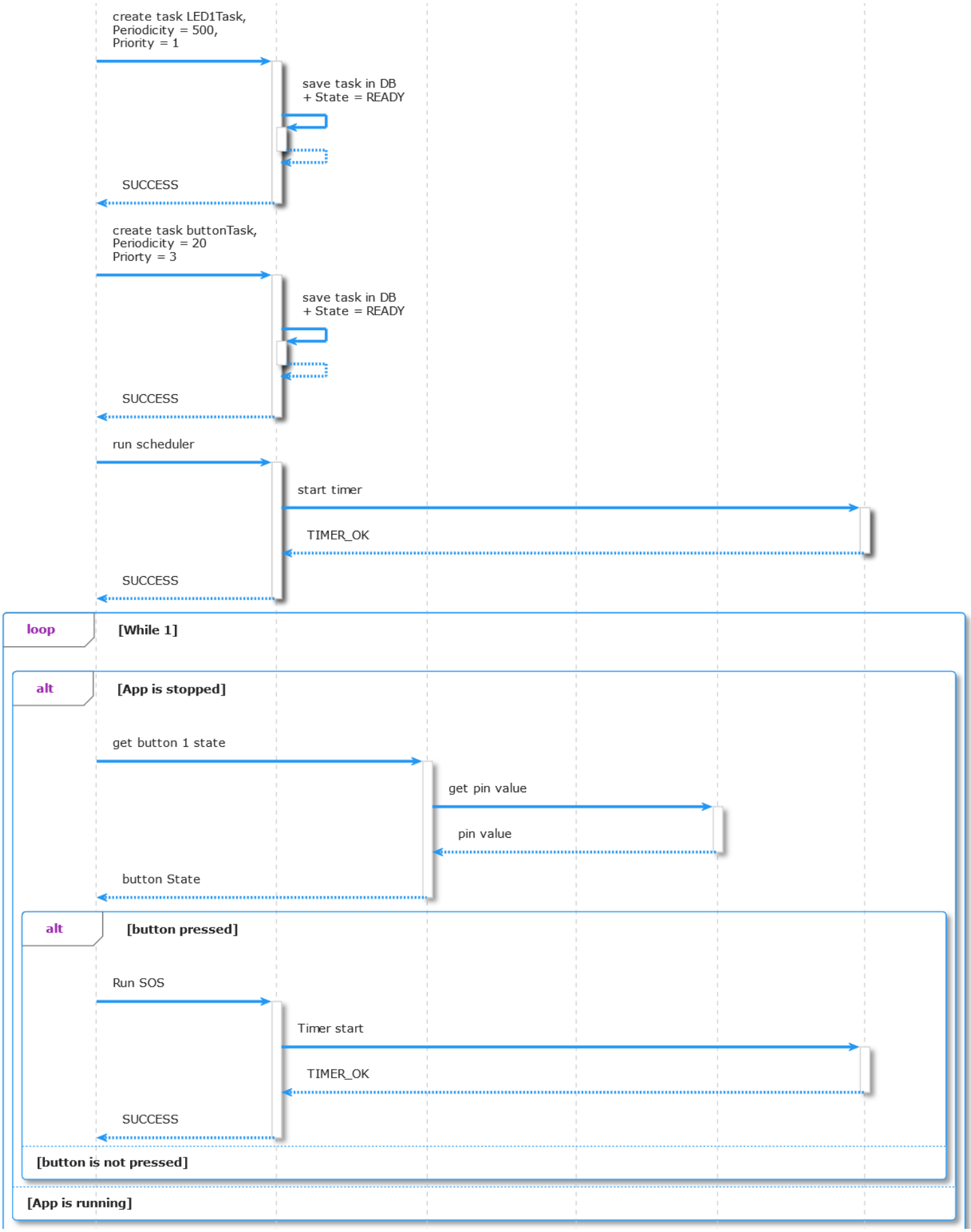
2.2. Sequence Diagram

Full Sequence Diagram Link

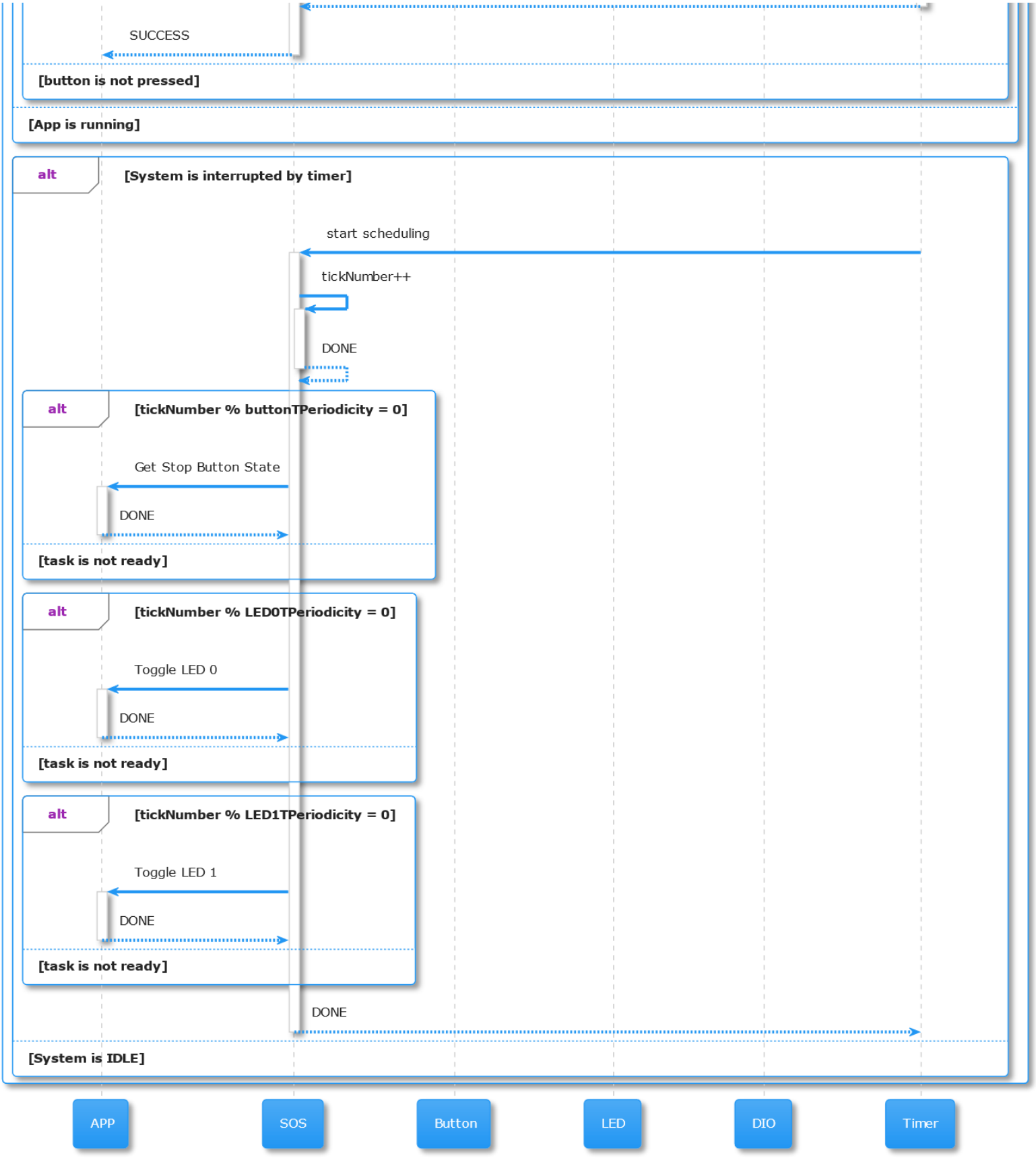
PART 1.



PART 2

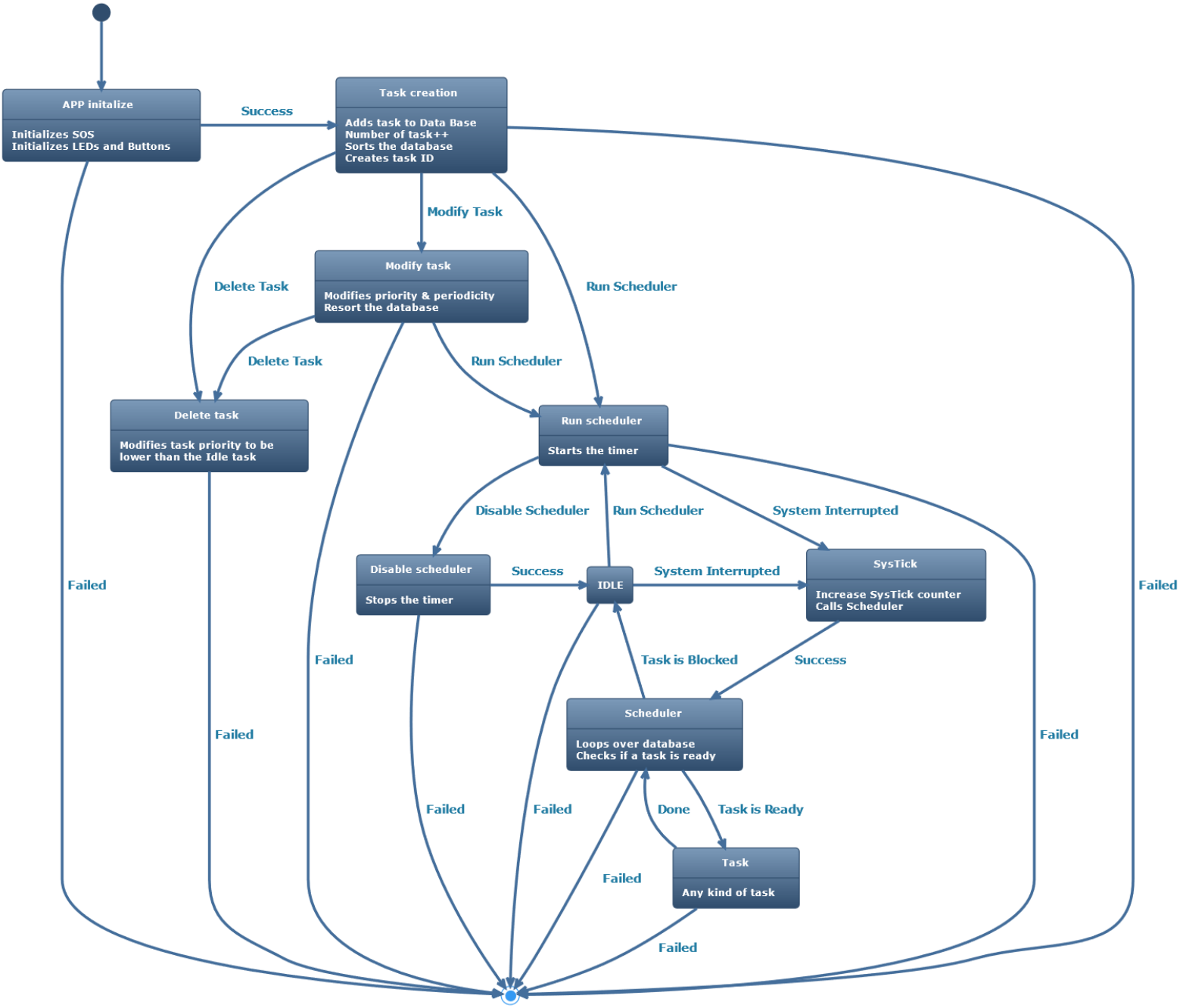

create task LED1Task,
Periodicity = 500,
Priority = 1
save task in DB
+ State = READY
SUCCESS
create task buttonTask,
Periodicity = 20
Priorty = 3
save task in DB
+ State = READY
SUCCESS
run scheduler
start timer
TIMER_OK
SUCCESS
loop
[While 1]
alt
[App is stopped]
get button 1 state
get pin value
pin value
button State
alt
[button pressed]
Run SOS
Timer start
TIMER_OK
SUCCESS
[button is not pressed]
[App is running]

PART 3

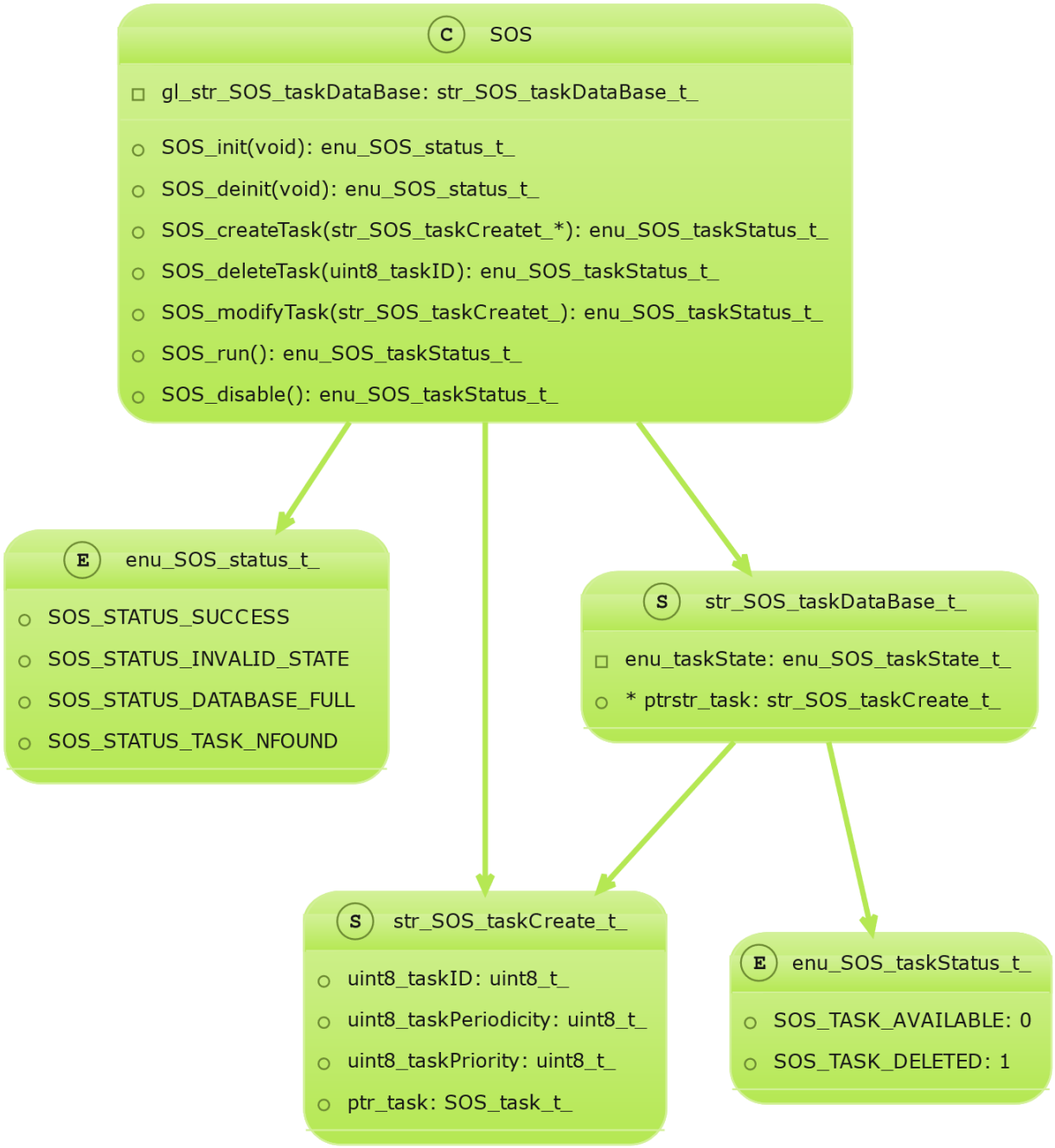

SUCCESS
[button is not pressed]
[App is running]
alt
[System is interrupted by timer]
start scheduling
tickNumber++
DONE
alt
[tickNumber % buttonTPeriodicity = 0]
Get Stop Button State
DONE
[task is not ready]
alt
[tickNumber % LED0TPeriodicity = 0]
Toggle LED 0
DONE
[task is not ready]
alt
[tickNumber % LED1TPeriodicity = 0]
Toggle LED 1
DONE
[task is not ready]
DONE
[System is IDLE]
APP
SOS
Button
LED
DIO
Timer

2.3. State Machine Diagram

Link for a better version HERE



2.4. Class Diagram



2.5. Modules Description

2.5.1. DIO (Digital Input/Output) Module

The DIO driver is responsible for reading input signals from the system's sensors (such as buttons) and driving output signals to the system's actuators (such as LEDs). It provides a set of APIs to configure the direction and mode of each pin (input/output, pull-up/down resistor), read the state of an input pin, and set the state of an output pin.

2.5.2. Timer

The Timer driver is responsible for initializing the timer peripheral, it can be configured to choose which timer will be working on which mode and other important configurations for the timer, but mainly I developed the timer to be working in normal mode only and at the same time it can generate any kind of output using normal mode.

2.5.3. LED

The LED driver is used to initialize LEDs used as output and control them as it can turn them on or off or toggle them.

2.5.4. Button

The button driver is responsible for initializing the buttons used as inputs and control them as it gets the button state

2.5.5. SOS

The SOS driver is used for initializing Timer used for the operating system, for initializing tick time too, for creating the database for tasks and finally for scheduling the tasks.