EIC

Embedded Real-Time systems

Assignment 3 - Modeling and implementaion of a system behavior using UML patterns.

Name	Initials	student number
Jakob Bjerregaard Olsen	JBO	202109022
Alexander Nygaard Thomsen	AT	201911329
Wouter Oudijk 6	WO	202109059

Lecturer: Jalil Boudjadar, Kim Bjerge

Due date: 01/12/2025

1 Introduction

The main goal is this assignment is to implement a simulated version of an embedded system using state patterns, more specifically a concurrent state machine, where the handling of event are processed, by using *GOF State pattern*, Singletong pattern and the *Active object pattern*

In this assignment, the application has been implemented by adhering to these requirements, and following the provided state diagram given with the assignment. Some liberty has been taken in terms of implementation, regarding how to interact with the system, but also how the states and classes has been used and implemented. The goal of this report is to explain the major outlines of the decisions made along, with visual descriptions and depictions. This includes large code snippets and UML diagrams which will serve as context.

Included with this report is the full code implementation, an executeable file, all UML diagrams and code to recreate/recompile the diagrams.

1.1 tools used

To complete this assignment some tools have been used.

Tool	**Description**
VS Code	IDE, for programming the assignemt
C++	the programming language required to implement the assignment and a compiler for C++14++ is required.
LaTeX	using MIKTEX
plantUML	UML tool of choice, converting it to a programming language, requires Java JDK
Git	for version control, using GITHUB to control the repo

Table 1: Tools used to complete the assignment

1.2 Goals

- **GOF Singleton pattern** To ensure the correct behavior of the statemachine, the Singleton pattern must be used, it ensures that only one instance of every object class can exist at any given time.
- **GOF state pattern** This pattern ensures that an object changes behaviour when it changes state, but to the outside world it looks like a new object. Instead of enumeration, the state and its behaviour is encapsulated in multiple classes, where our object can delegate to the current state class.
- **Active Object pattern** The main purpose of the active object pattern, is it separate method invocation from method execution, by decoupling. This lets us asynchronously invoke methods while not tying up valuable time to execute those methods, by having separate threads do both scheduling and execution.
  - Active object pattern must implement the provided interface
  - not be blocking
- **follow the provided state pattern** With the assignment, the *EmbeddedSystemX* class was provided, which outlined the overall requirement for the application.

2 Solution

2.1 Overall architecture

3 Implementation