Pressure Controller Project

Mastering Embedded System online Diploma

By

Abdelrahman El-Badawy

Under the guidance of

Eng.Keroles Shenouda

Department of Computer & Systems Engineering,

FEHU,Helwan

October, 2023

ABSTRACT

This project is centered around the development of a pressure control system aimed at monitoring and managing cabin pressure within a controlled environment. The client's specifications mandate the implementation of an alarm triggered when the cabin pressure exceeds 20 bars, with a subsequent alarm duration of 60 seconds. The project encompasses various stages, starting with a requirements diagram to delineate the system's needs. Following this, a comprehensive system analysis involves the creation of key diagrams such as the Use Case Diagram, Activity Diagram, and Sequence Diagram. These diagrams will provide insights into the system's functionalities, interactions, and behavior. In the subsequent stage of system design, modules with their respective state machines will be outlined, ensuring a structured and efficient implementation. The primary objective is to construct a robust pressure control system that accurately detects pressure fluctuations and promptly initiates alarms, prioritizing the safety and well-being of the cabin's occupants. This document provides an overview of the project, presenting the client's specifications and outlining the proposed approach to meet the defined requirements through meticulous system analysis and design.

CONTENTS

LIST OF FIGURES

Chapter 1

REQUIREMENTS DIAGRAM

REQUIREMENTS DIAGRAM

The Requirements Diagram provides a structured representation of the system's requirements, which include both functional and non-functional aspects. Functional requirements define the specific functions, features, or actions the system must perform, while non-functional requirements encompass qualities the system should possess, such as performance, usability, or reliability.

The diagram organizes these requirements hierarchically, illustrating the relationships and dependencies between them. At the top level, it outlines the major system goals or objectives. These objectives are then decomposed into more detailed requirements at lower levels, allowing for a clear understanding of the system's scope and the functionalities it needs to support.

By creating a Requirements Diagram, the project team gains a visual map of the client's expectations and system necessities. This visual representation aids in traceability, ensuring that every aspect of the system aligns with the specified requirements. It sets the foundation for subsequent phases of the project, guiding the system analysis, design, and development efforts.

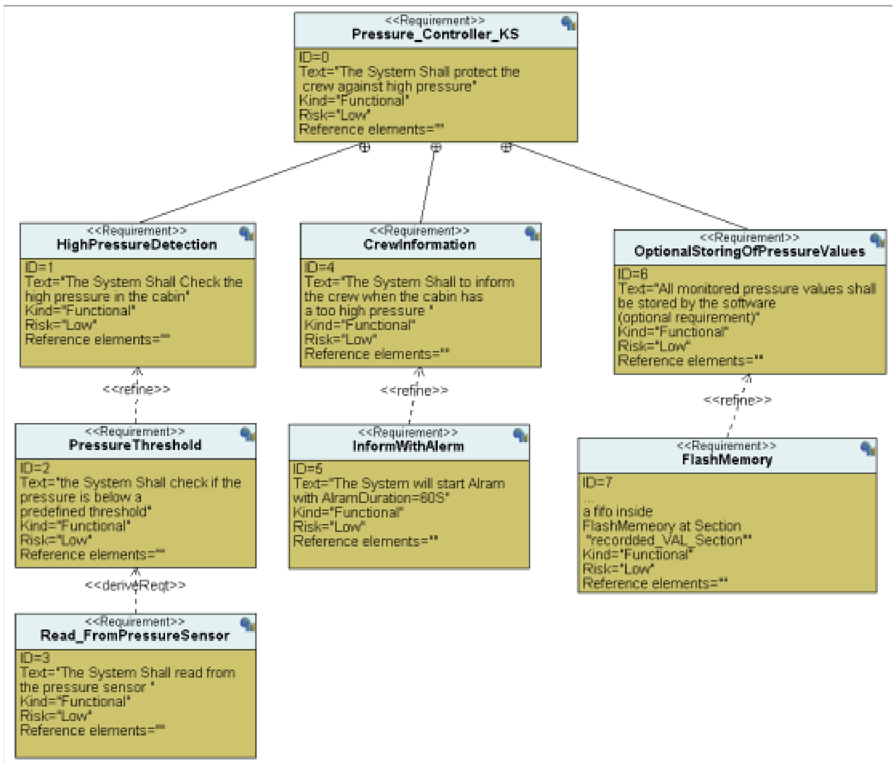


Figure 1.1: Requirements Diagram

Chapter 2

SYSTEM ANALYSIS

The System Analysis phase delves into a comprehensive exploration and understanding of the pressure control system's functionalities and interactions. It aims to uncover system requirements, user interactions, and the flow of activities within the system. Three crucial diagrams are employed for this purpose:

2.0.1 USE CASE DIAGRAM

The Use Case Diagram provides a high-level view of the system's functionalities from a user's perspective. It illustrates the various actions or use cases that the system offers to its users (actors), showcasing the interaction between actors and the system.

2.0.2 ACTIVITY DIAGRAM

The Activity Diagram focuses on the dynamic aspects of the system, capturing the flow of activities and actions within the system. It details the sequential steps involved in achieving specific goals and highlights decision points and possible alternate paths.

2.0.3 SEQUENCE DIAGRAM

The Sequence Diagram offers a dynamic view of the system's behavior by illustrating the interactions between different entities (actors or components) over time. It provides a visual representation of the message flow and the order of interactions during specific scenarios.

2.1 USE CASE :

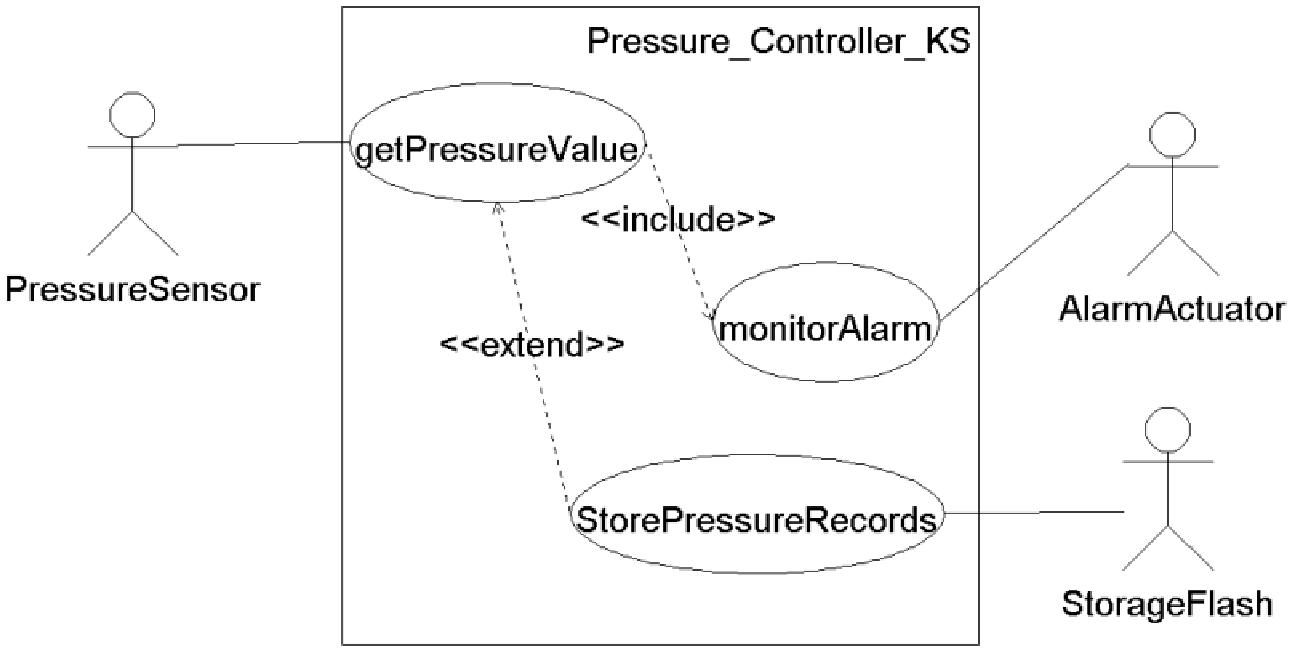


Figure 2.1: Use Case Diagram

2.2 ACTIVITY DIAGRAM :

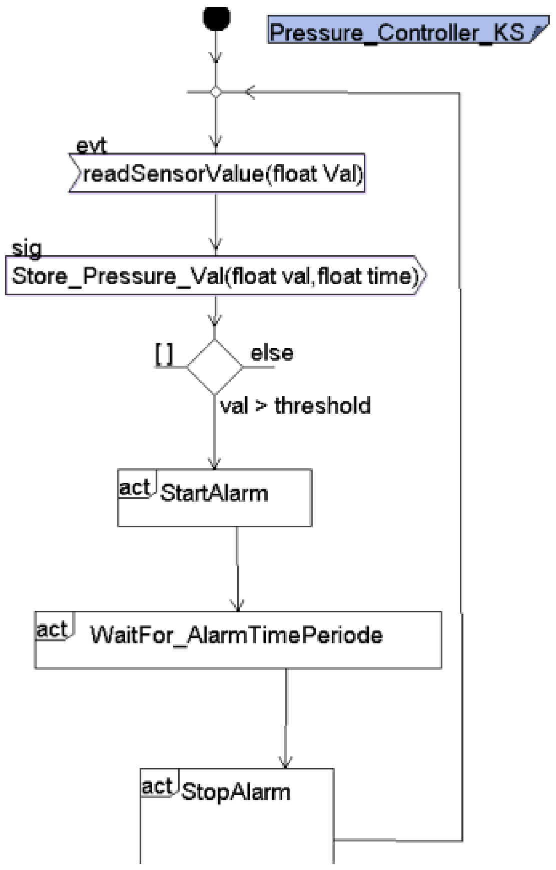

Pressure_Controller_KS
evt
readSensorValue(float Val)
sig
Store_Pressure_Val(float val,float time)
[]
else
val > threshold
act StartAlarm
act WaitFor_AlarmTimePeriode
act StopAlarm

Figure 2.2: Activity Diagram:

2.3 SEQUENCE DIAGRAM :

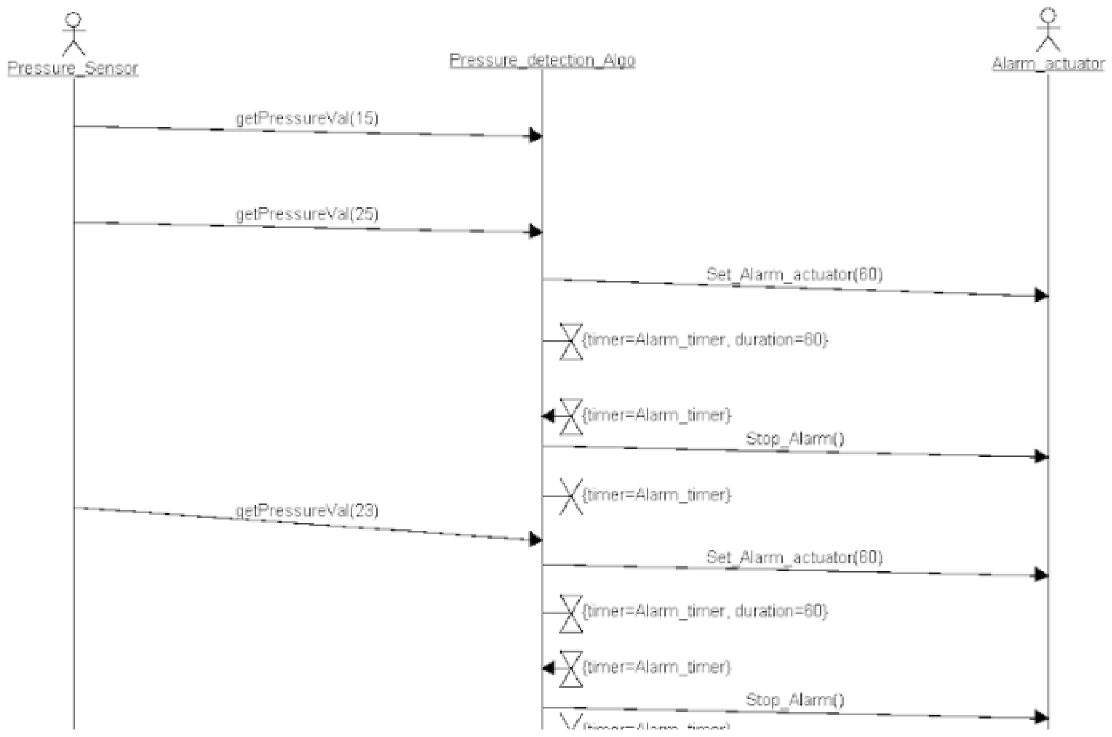


Figure 2.3: Sequence Diagram

Chapter 3

SYSTEM DESIGN (MODULES WITH ITS OWN STATE MACHINES)

SYSTEM DESIGN (MODULES WITH THEIR OWN STATE MACHINES)

The System Design phase is a critical stage in the project, focusing on the architectural blueprint and internal structure of the pressure control system. This phase aims to organize the system's components into modules, each with its own well-defined responsibilities and behaviors. Moreover, it involves the creation of state machines, a fundamental tool to model and visualize the behavior of each module.

3.0.1 MODULES

In this phase, the system is broken down into distinct modules, encapsulating specific functionalities. Modules are units of code that are designed to be relatively independent, promoting modularity, reusability, and maintainability. Each module is responsible for a set of related tasks and features, allowing for efficient development and management of the entire system.

3.0.2 STATE MACHINES

State machines are pivotal in modeling the behavior of modules within the system. A state machine defines the various states a module can exist in, the events or triggers that cause state transitions, and the actions associated with each transition. This formalized representation aids in understanding the module's behavior in response to different inputs and conditions.

By employing state machines for each module, we ensure a structured and predictable behavior of the system. The transitions between states are clearly defined, enabling comprehensive testing and validation. Additionally, state machines facilitate communication and collaboration within the development team, ensuring a shared understanding of the module's functionality.

The integration of modules and their respective state machines forms the basis for the subsequent development phase, guiding the implementation process and providing a roadmap for creating a robust and functional pressure control system.

3.1 SYSTEM BLOCKS :

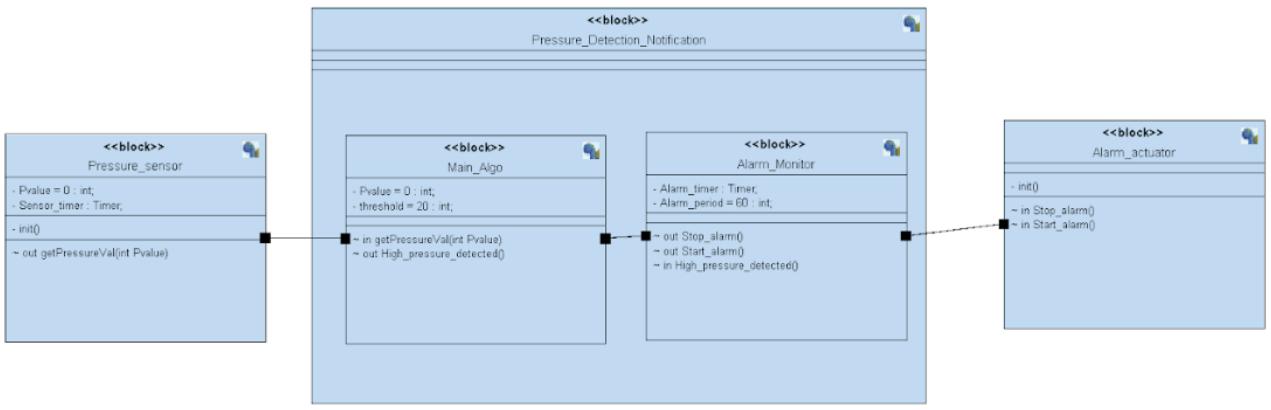


Figure 3.1: System Blocks Diagram

3.2 PRESSURE SENSOR DRIVER STATE :

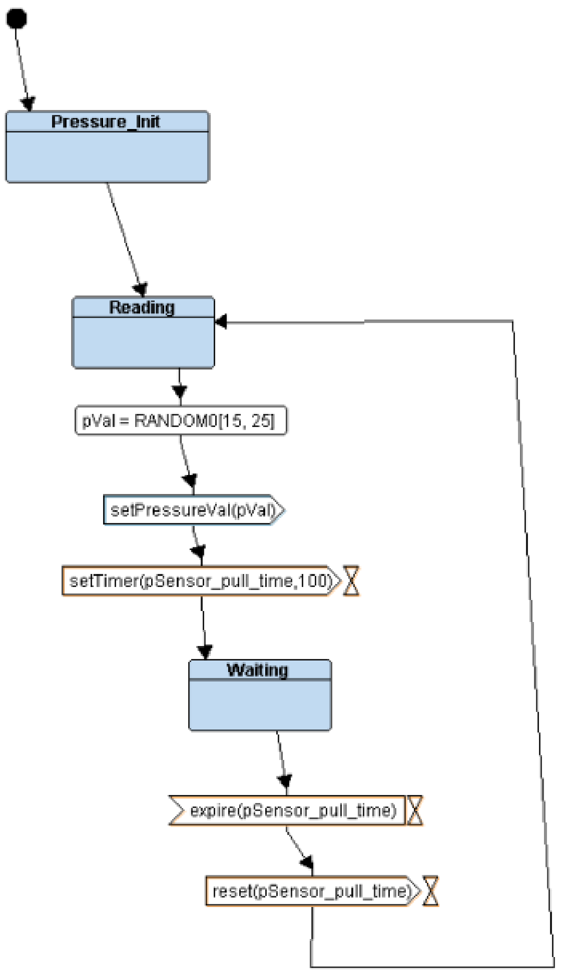


Figure 3.2: Pressure Sensor Driver State

3.3 MAIN ALGORITHM STATE :

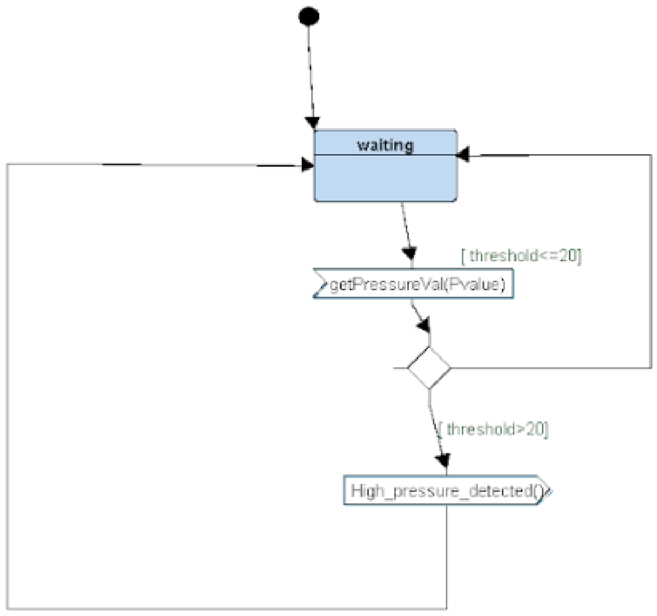


Figure 3.3: Main Algorithm State

3.4 ALARM MONITOR STATE :

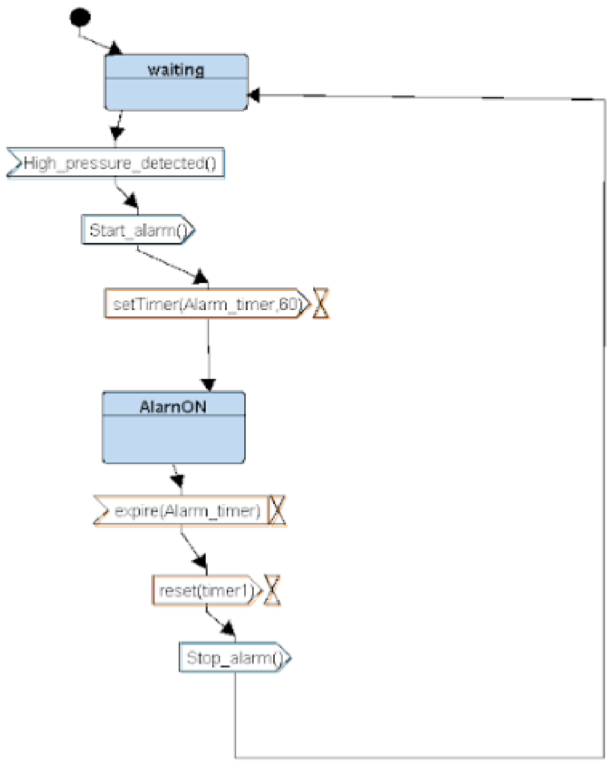


Figure 3.4: Alarm Monitor State

3.5 ALARM ACTUATOR DRIVER STATE :

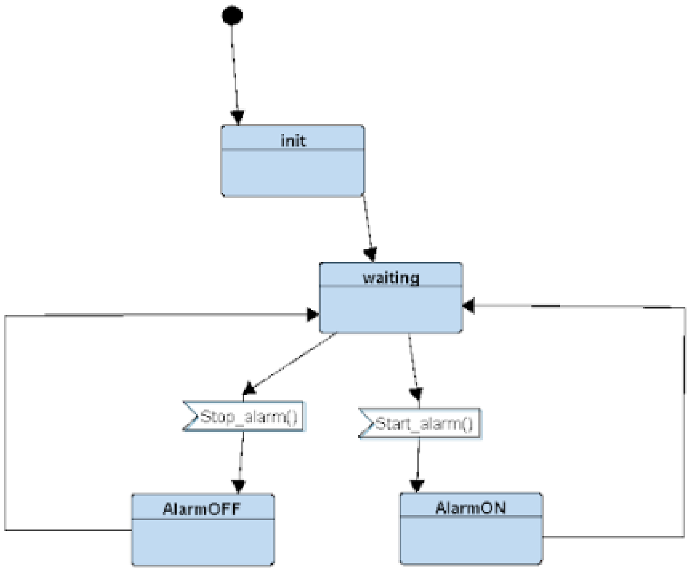


Figure 3.5: Alarm Actuator Driver State

Chapter 4

SOURCE CODE

4.1 GITHUB REPO

You can find on my GitHub The Pressure sensor: Pressure sensor

You can find on my GitHub The Main Algorithm: Main Algo

You can find on my GitHub The Alarm Monitor: Alarm Monitor

You can find on my GitHub The Alarm actuator: Alarm actuator

You can find on my GitHub The Integration: Integration

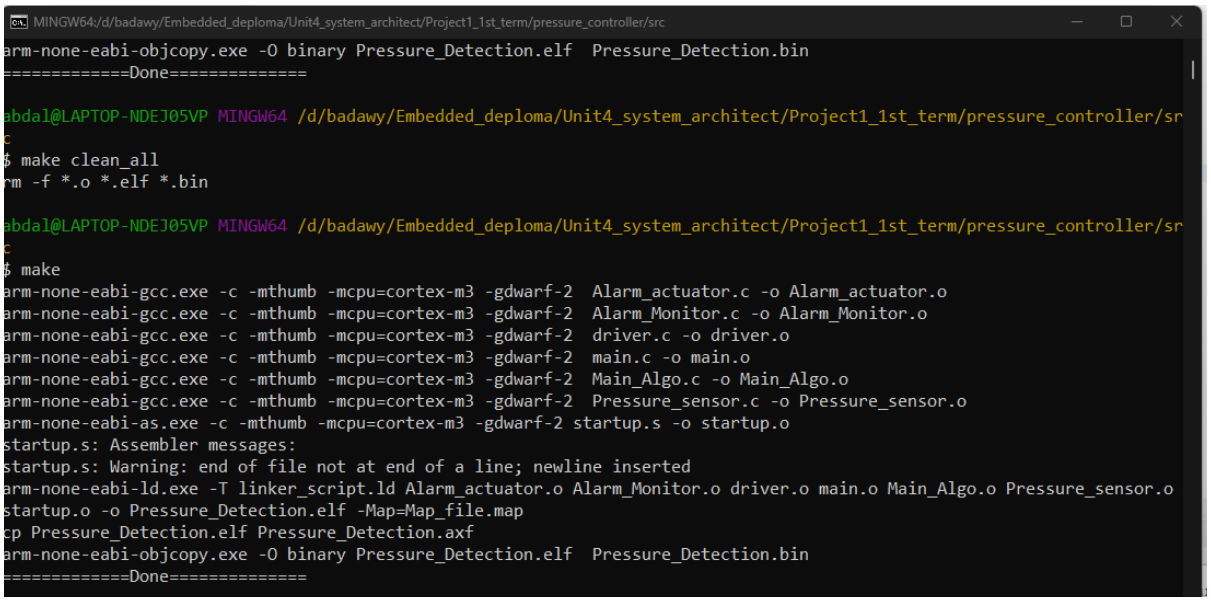
4.2 THE SOURCE CODE

The Src : GitHub

Chapter 5

SIMULATION RESULTS AND DISCUSSIONS

5.1 BUILDING THE PROJECT :


MINGW64:/d/badawy/Embedded_deploma/Unit4_system_architect/Project1_1st_term/pressure_controller/src

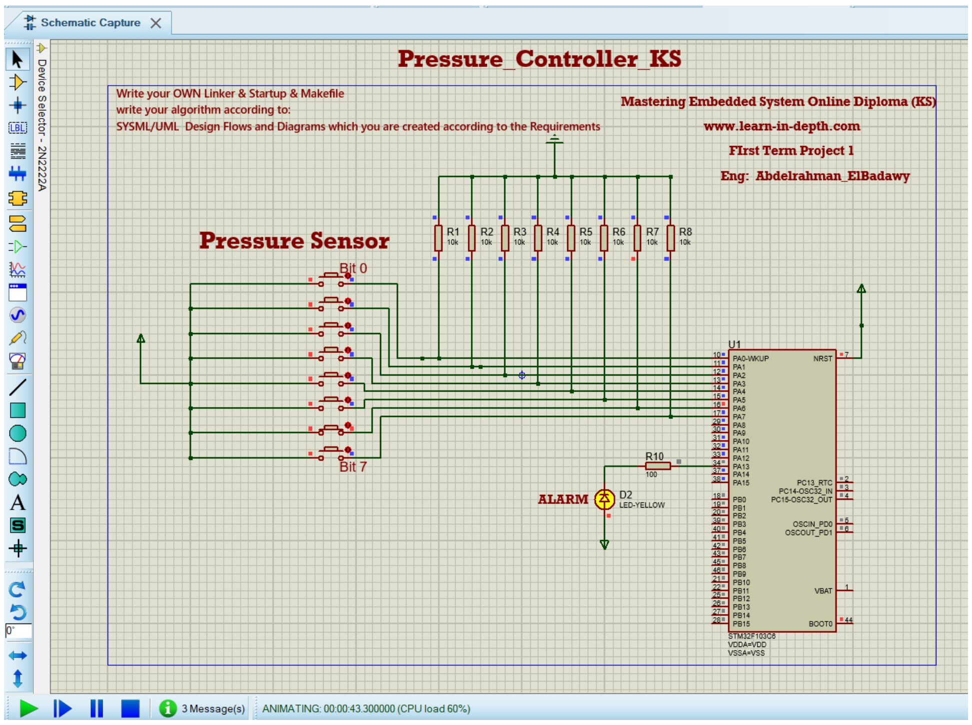
```
arm-none-eabi-objcopy.exe -O binary Pressure_Detection.elf  Pressure_Detection.bin
============Done=============

abdal@LAPTOP-NDEJ05VP MINGW64 /d/badawy/Embedded_deploma/Unit4_system_architect/Project1_1st_term/pressure_controller/sr
c
$ make clean_all
rm -f *.o *.elf *.bin

abdal@LAPTOP-NDEJ05VP MINGW64 /d/badawy/Embedded_deploma/Unit4_system_architect/Project1_1st_term/pressure_controller/sr
c
$ make
arm-none-eabi-gcc.exe -c -mthumb -mcpu=cortex-m3 -gdwarf-2  Alarm_actuator.c -o Alarm_actuator.o
arm-none-eabi-gcc.exe -c -mthumb -mcpu=cortex-m3 -gdwarf-2  Alarm_Monitor.c -o Alarm_Monitor.o
arm-none-eabi-gcc.exe -c -mthumb -mcpu=cortex-m3 -gdwarf-2  driver.c -o driver.o
arm-none-eabi-gcc.exe -c -mthumb -mcpu=cortex-m3 -gdwarf-2  main.c -o main.o
arm-none-eabi-gcc.exe -c -mthumb -mcpu=cortex-m3 -gdwarf-2  Main_Algo.c -o Main_Algo.o
arm-none-eabi-gcc.exe -c -mthumb -mcpu=cortex-m3 -gdwarf-2  Pressure_sensor.c -o Pressure_sensor.o
arm-none-eabi-as.exe -c -mthumb -mcpu=cortex-m3 -gdwarf-2 startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline inserted
arm-none-eabi-ld.exe -T linker_script.ld Alarm_actuator.o Alarm_Monitor.o driver.o main.o Main_Algo.o Pressure_sensor.o
startup.o -o Pressure_Detection.elf -Map=Map_file.map
cp Pressure_Detection.elf Pressure_Detection.axf
arm-none-eabi-objcopy.exe -O binary Pressure_Detection.elf  Pressure_Detection.bin
============Done=============
```

Figure 5.1: Building the project

5.2 ALARM ON :


Pressure_Controller_KS

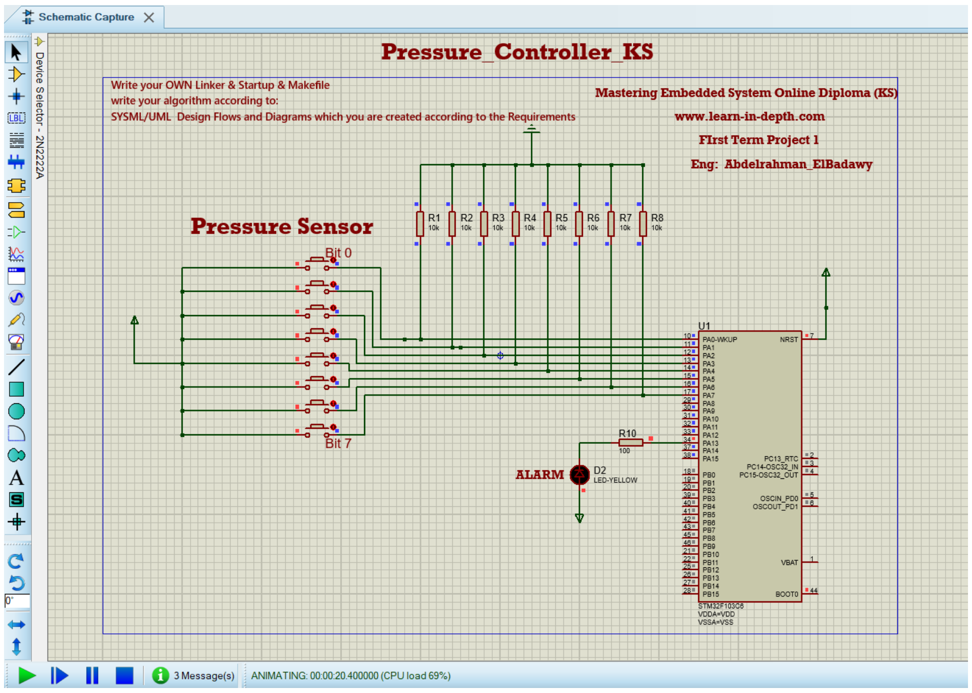
Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements

Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Eng: Abdelrahman_ElBadawy

3 Message(s) ANIMATING: 00:00:43.300000 (CPU load 60%)

Figure 5.2: Alarm is On

5.3 ALARM OFF :


Pressure_Controller_KS

Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements

Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Eng: Abdelrahman_ElBadawy

3 Message(s) ANIMATING: 00:00:20.400000 (CPU load 69%)

Figure 5.3: Alarm is Off