MASTERING EMBEDDED SYSTEM ONLINE DIPLOMA

www.learn-in-depth.com

FIRST TERM - PROJECT 1

ENG. AHMED ESSAM ELDIN ELSHAFIE

https://www.learn-in-depth.com/online-diploma/ahmedessam020@gmail.com

Table of Contents

Table of Figures

PRESSURE DETECTION SYSTEM

1 INTRODUCTION

This project is mainly about implementing a pressure detection system. Its main purpose is to fire an alarm whenever the measured pressure inside the crew cabin has passed the predefined threshold to notify the crew that the pressure is high. We will start by listing the requirements given by the client.

2 DESIGN SEQUENCE

2.1 Case Study

The client expects to be delivered a software system with the following requirements:

- A Pressure Detector to notify the crew cabin through an alarm that the pressure has exceeded 20 bars in the cabin
- The alarm should be on once fired for a time duration of 60 secs.
- Keep track of the pressure measured values.

After listing the client's requirements, we need to set our assumptions to avoid any further conflicts. Here is the list of our assumptions:

- The controller setup and shutdown procedures are not modeled.
- The controller maintenance is not modeled.
- The pressure sensor never fails.
- The alarm actuator never fails.
- The controller never faces power cut.

Also, we will inform the client that the requirement "Keep track of the pressure measured values" will not be modeled in the first version of the design. It could be implemented in a further version.

2.2 SDLC Model

Concerning the Software Development Life Cycle model, we decided to choose the V-Model in order to focus on the implementation phase at first then proceed to the validation & testing phase.

V-Model consists of the following phases:

- Business requirement specification.
- System requirement specification.
- High level design.
- Low level design.
- Coding.
- Unit testing.
- Component Testing.
- System Integration Testing.
- Acceptance Testing.

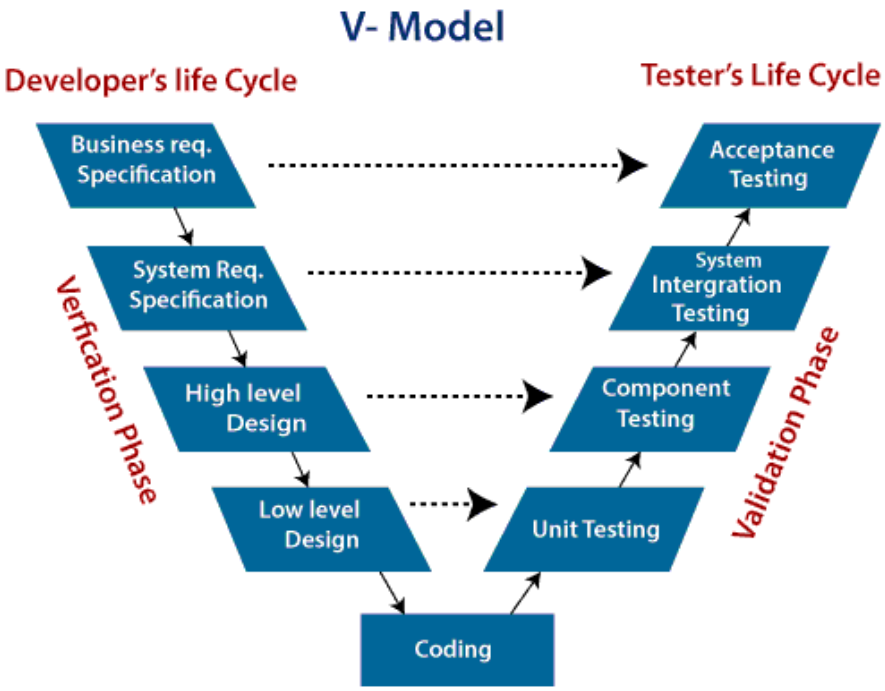


Figure 1 - SDLC V-Model

2.3 Requirement

In this section, we will focus on the requirements listed by the client. We need to design the requirement diagram in order to elaborate the given requirements and discuss their relations with the client to be confirmed.

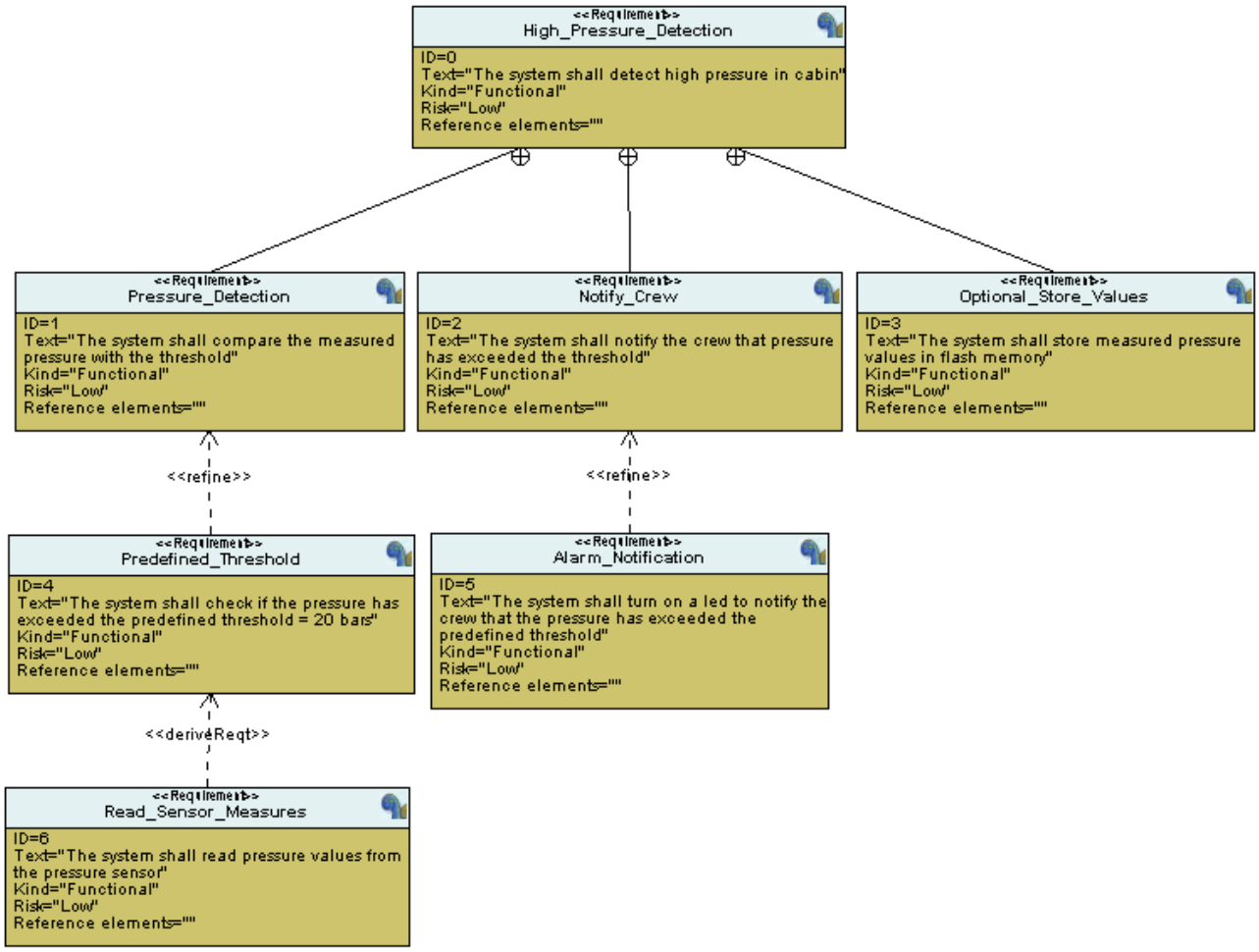


Figure 2 - Requirement Diagram

2.4 System Exploration/Partitioning

In this section, we should discuss which ECU-s we will use to develop this system. We decided that we will implement the following system using the STM32F103F6 microcontroller.

This microcontroller has ARM 32-bits Cortex M3 processor.

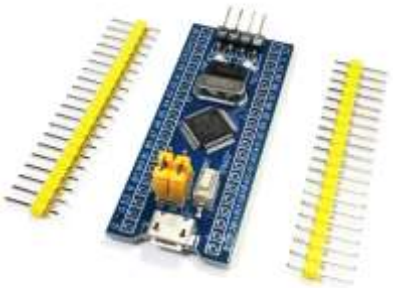


Figure 3 - STM32F103C6

The following is the specs of this microcontroller used to implement this software system:

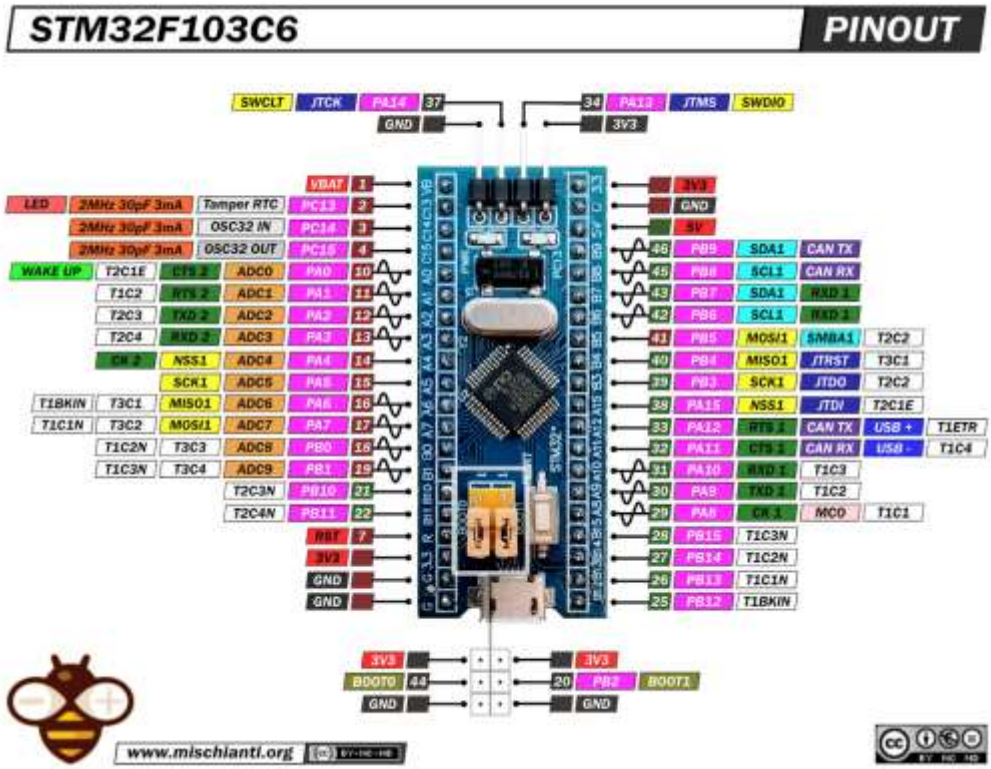


Figure 4 - STM32F103C6 Specifications

2.5 System Analysis

In this section, we will design the main 3 UML diagrams that elaborates the flow of this software system. The UML diagrams to be designed are Use Case Diagram, Activity Diagram and Sequence Diagram.

2.5.1 Use Case Diagram

This diagram shows the main actors in the system and each of their responsibility cases.

Main actors:

- Pressure Sensor.
- Alarm Actuator.
- Flash Memory.

Main Responsibilities:

- Comparing Algorithm.
- Get Pressure Value.
- Monitor Alarm.
- Store Pressure Value.

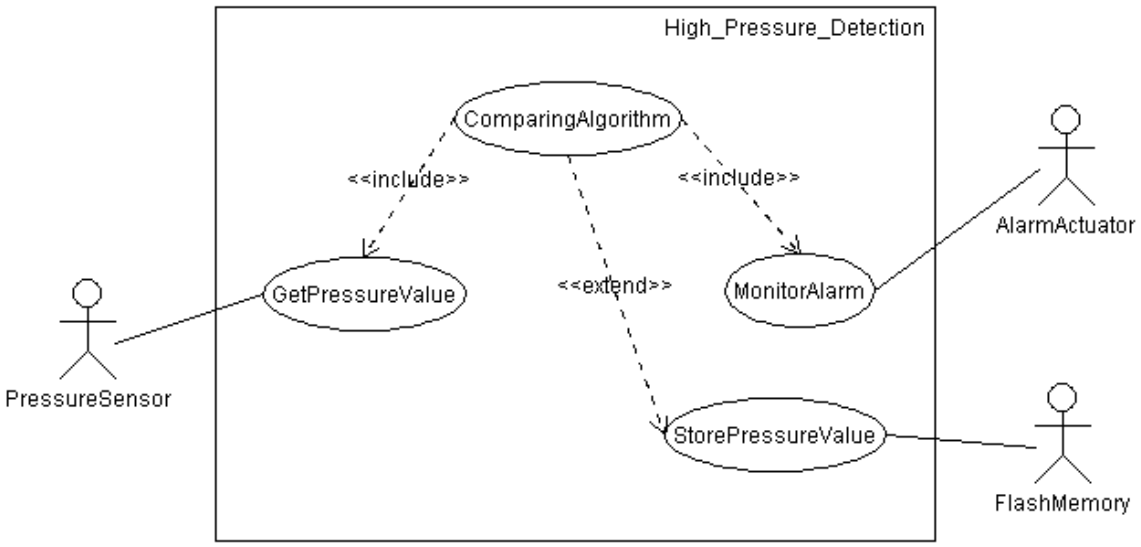


Figure 5 - Use Case Diagram

2.5.2 Activity Diagram

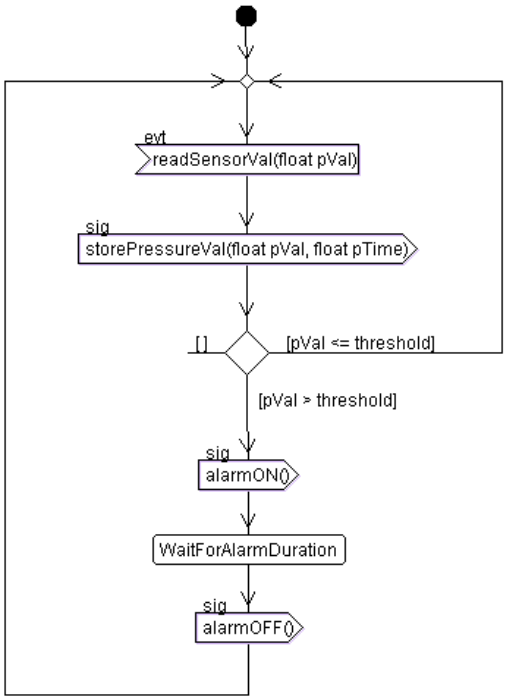


Figure 6 - Activity Diagram

2.5.3 Sequence Diagram

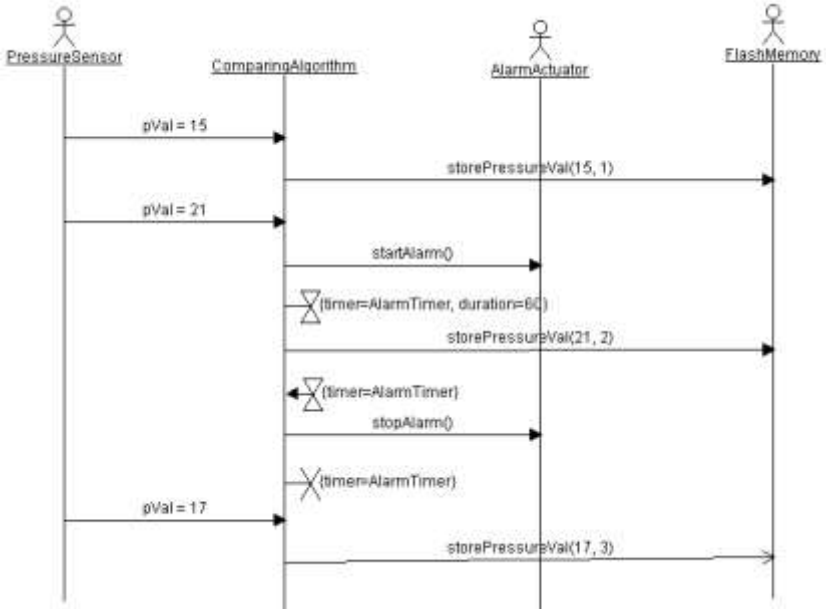


Figure 7 - Sequence Diagram

2.6 System Design

Now, after finishing the system analysis phase we should proceed to the system design phase. In this phase, we should design the main block diagram that will contain the modules to be implemented with the relations between each module followed by the state diagrams for each module to elaborate their flow.

2.6.1 Block Diagram

As our main modules, we have HighPressureDetection module with 2 sub-modules ComparingAlgorithm and AlarmMonitor. Another two modules which will be PressureSensorDriver and AlarmActuatorDriver.

PressureSensorDriver:

- Attributes: pVal, pTimer.
- Methods: pDriver_init().
- Signals: out > getPressureVal(int pVal).

ComparingAlgorithm:

- Attributes: pVal, threshold.
- Signals: in > getPressureVal(int pVal), out > highPressureFlag().

AlarmMonitor:

- Attributes: alarmDuration, aTimer.
- Signals: in > highPressureFlag(), out > alarmON(), out > alarmOFF().

AlarmActuatorDriver:

- Methods: aDriver_init().
- Signals: in > alarmON(), in > alarmOFF().

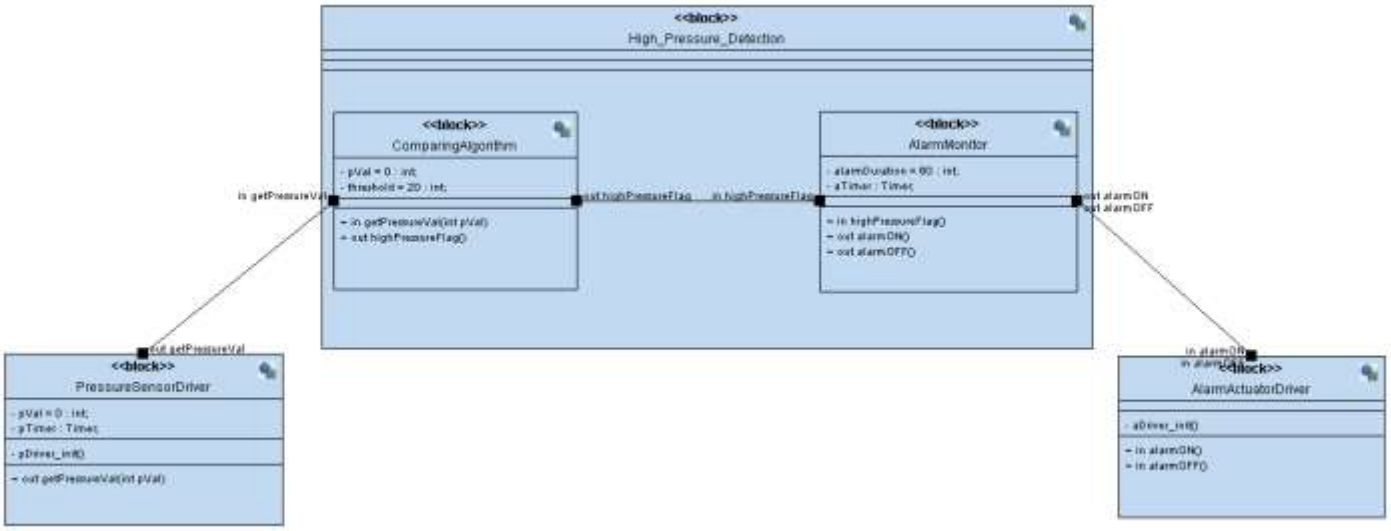


Figure 8 - Block Diagram

2.6.2 State Machines Diagram

2.6.2.1 PressureSensorDriver State Machine

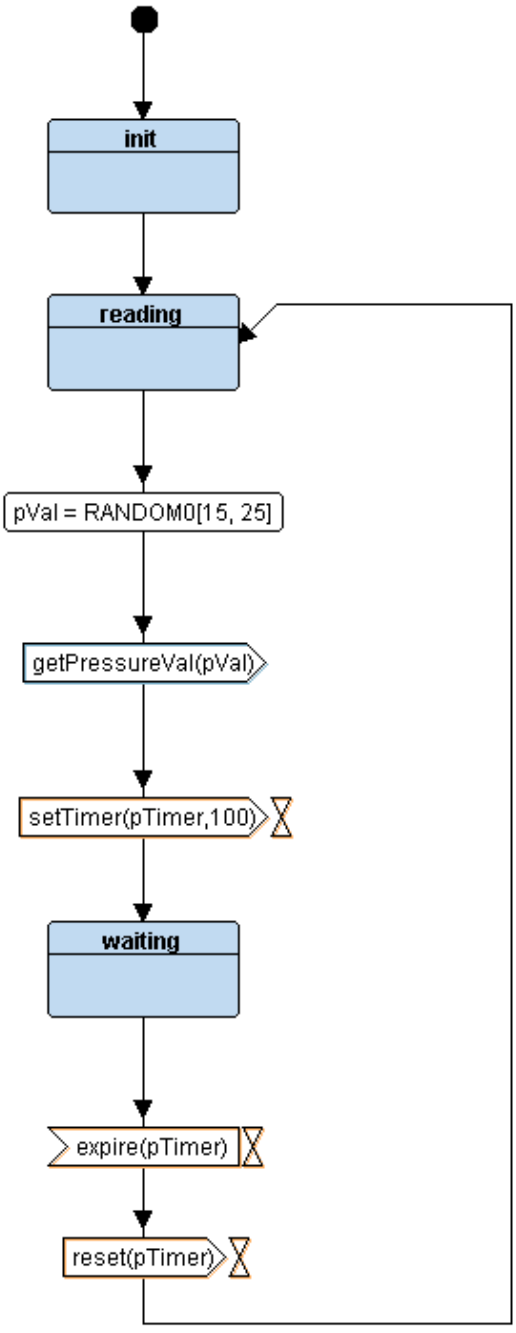


Figure 9 - PressureSensorDriver State Machine

2.6.2.2 ComparingAlgorithm State Machine

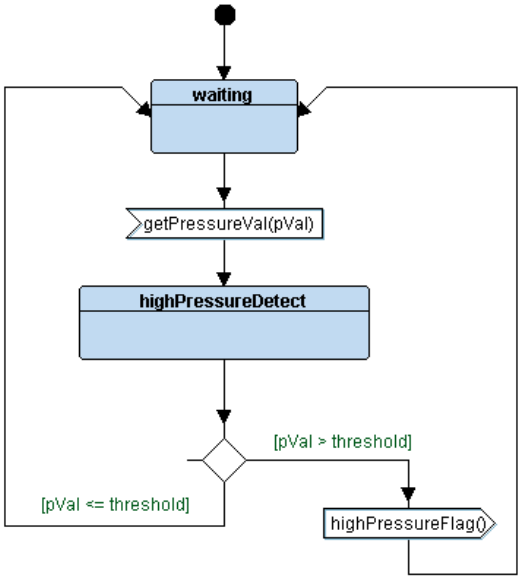


Figure 10 - ComparingAlgorithm State Machine

2.6.2.3 AlarmMonitor State Machine

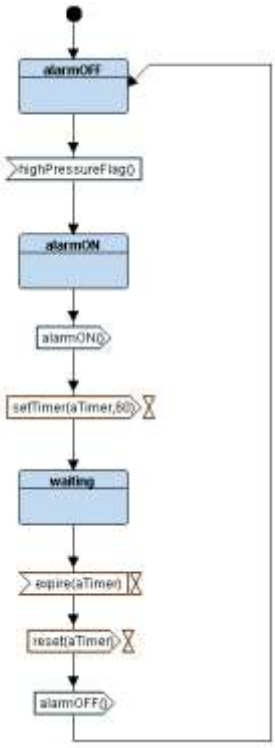


Figure 11 - AlarmMonitor State Machine

2.6.2.4 AlarmActuatorDriver State Machine

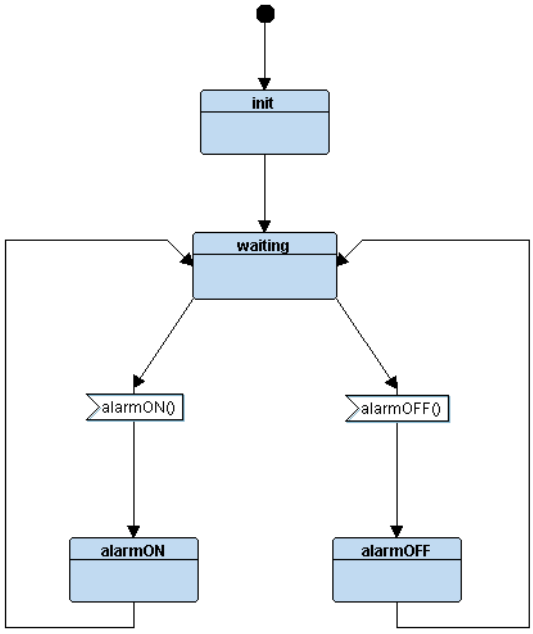


Figure 12 - AlarmActuatorDriver State Machine

2.6.3 State Machine Simulation

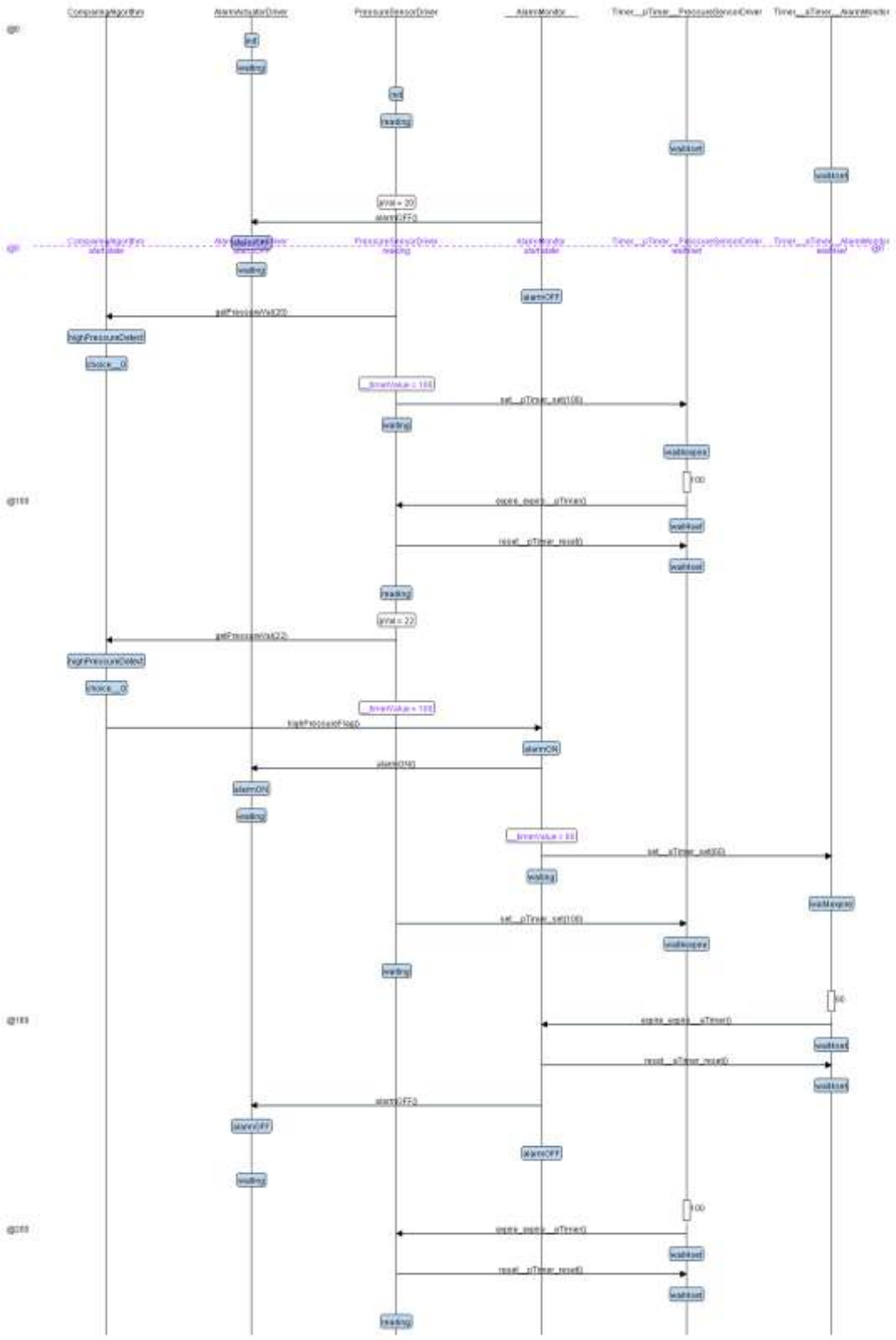


Figure 13 - State Machine Simulation

3 SOFTWARE IMPLEMENTATION

In this section, we will implement the software system according to the designed state machines considering their flow.

3.1 Source Code

3.1.1 PressureSensorDrive.h

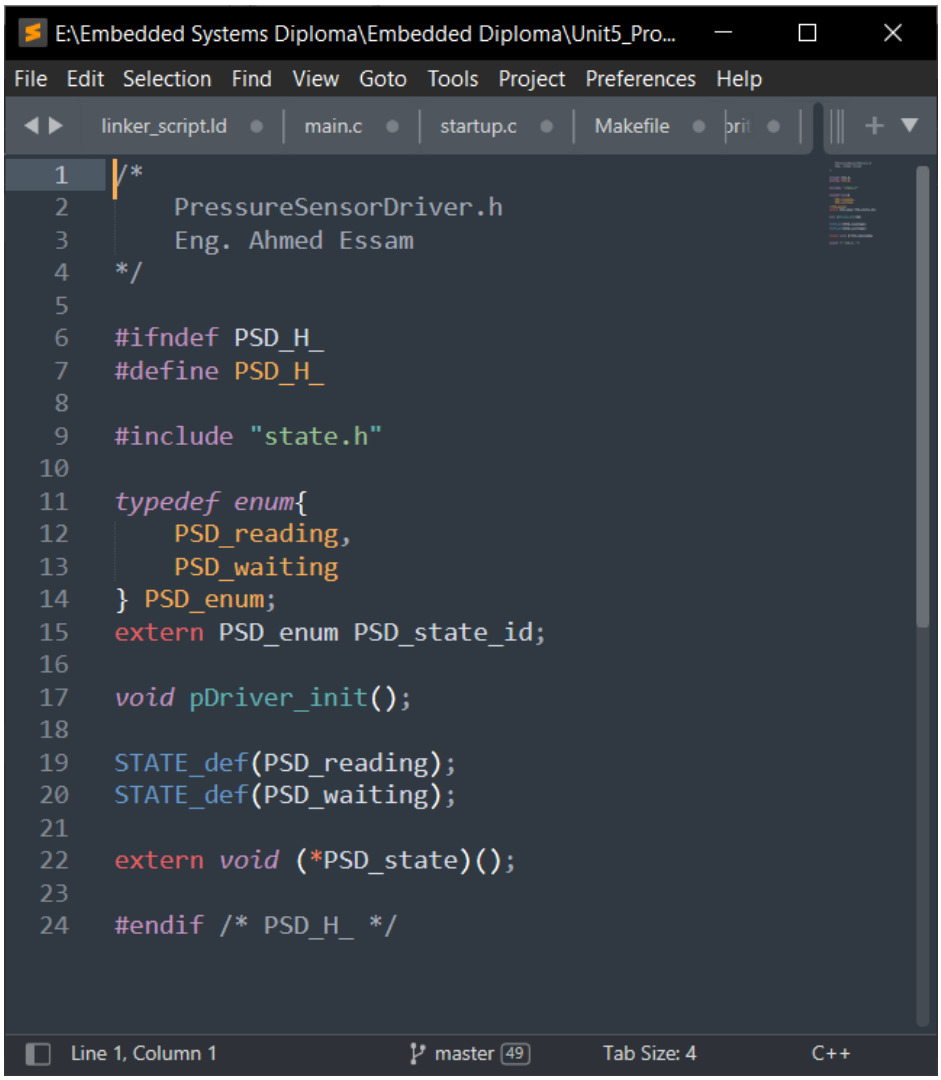


Figure 14 - PressureSensorDrive.h

3.1.2 PressureSensorDrive.c

Figure 15 - PressureSensorDrive.c

3.1.3 ComparingAlgorithm.h

E:\Embedded Systems Diploma\Embedded Diploma\Unit5_Pro...

File Edit Selection Find View Goto Tools Project Preferences Help

Pressur | PressureSensorDriver.c | ComparingAlgorithm.c | rithm.h

```
1  /*
2      ComparingAlgorithm.h
3      Eng. Ahmed Essam
4  */
5
6  #ifndef CA_H_
7  #define CA_H_
8
9  #include "state.h"
10
11 typedef enum{
12     CA_detect,
13     CA_waiting
14 } CA_enum;
15 extern CA_enum CA_state_id;
16
17 STATE_def(CA_detect);
18 STATE_def(CA_waiting);
19
20 extern void (*CA_state)();
21
22 #endif /* CA_H_ */
```

Line 1, Column 1 | master 49 | Tab Size: 4 | C++

Figure 16 - ComparingAlgorithm.h

3.1.4 ComparingAlgorithm.c

E:\Embedded Systems Diploma\Embedded Diploma\Unit5_Projects\First Ter...

File Edit Selection Find View Goto Tools Project Preferences Help

Pres | PressureSensorDriver.c | ComparingAlgorithm.c | ComparingAlgorithm.h

```c
/*
    ComparingAlgorithm.c
    Eng. Ahmed Essam
*/

#include "ComparingAlgorithm.h"

int CA_pVal = 0;
const int threshold = 20;

CA_enum CA_state_id;
void (*CA_state)();

void setPressureVal(int pVal){
    CA_pVal = pVal;
    CA_state = STATE(CA_detect);
}

STATE_def(CA_detect){

    //State Name
    CA_state_id = CA_detect;

    //State Action
    if(CA_pVal > threshold){
        highPressureFlag();
    }
    CA_state = STATE(CA_waiting);
}

STATE_def(CA_waiting){

    //State Name
    CA_state_id = CA_waiting;
}

```

Line 1, Column 1 | master 49 | Tab Size: 4 | C

Figure 17 - ComparingAlgorithm.c

3.1.5 AlarmMonitor.h

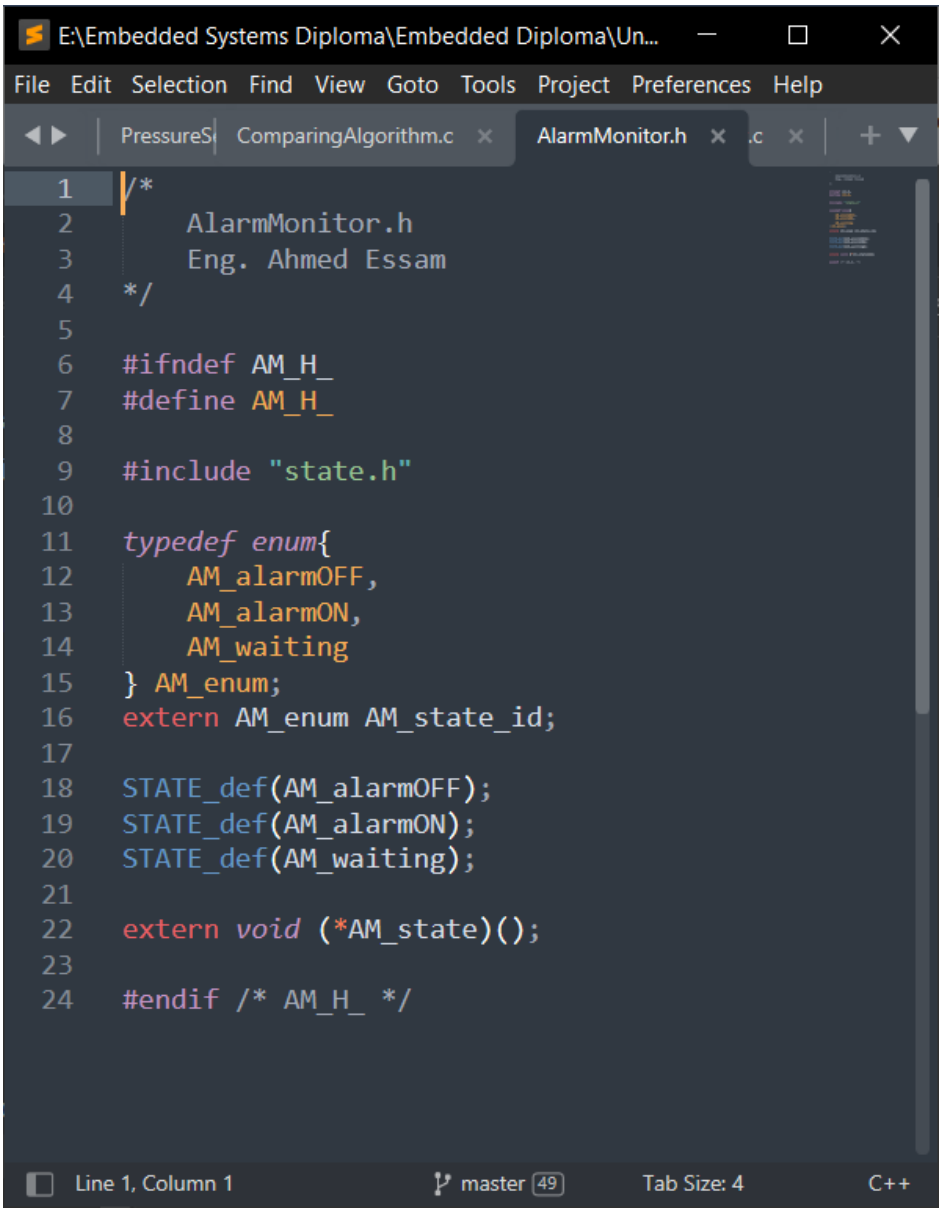


Figure 18 - AlarmMonitor.h

3.1.6 AlarmMonitor.c

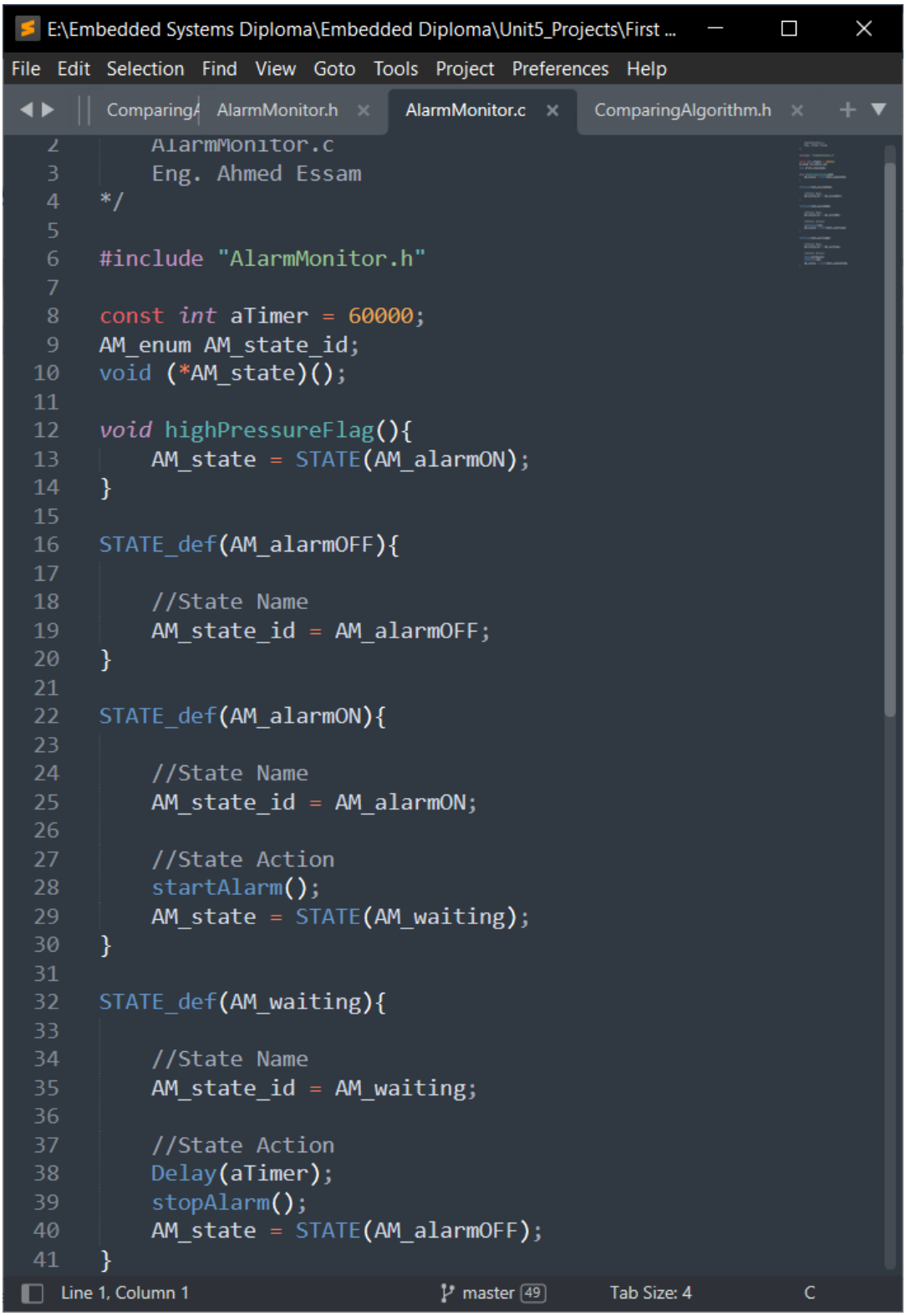


Figure 19 - AlarmMonitor.c

3.1.7 AlarmAcutatorDriver.h

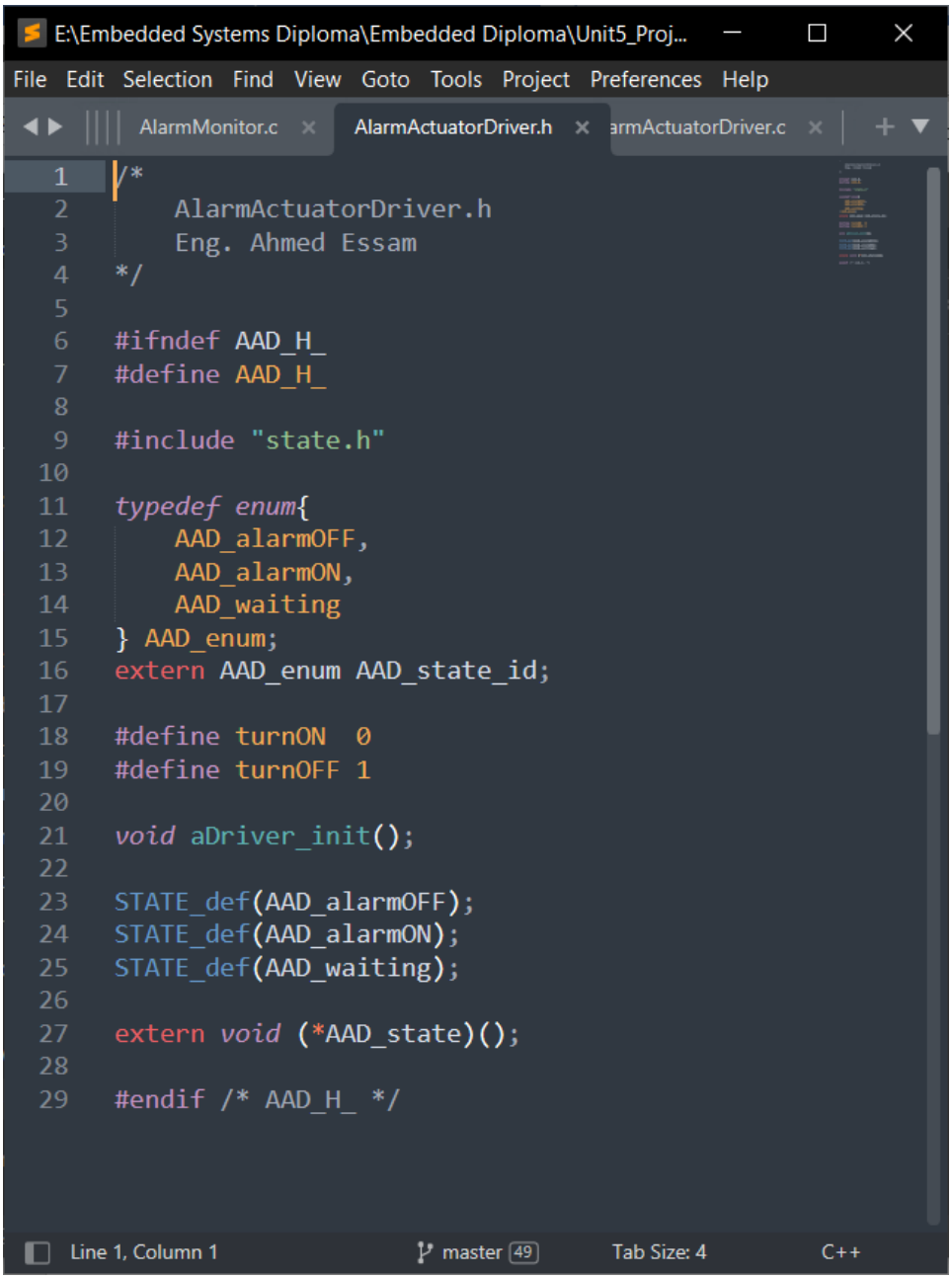


Figure 20 - AlarmAcutatorDriver.h

3.1.8 AlarmAcutatorDriver.c

Figure 21 - AlarmAcutatorDriver.c

3.1.9 driver.h

Figure 22 - driver.h

3.1.10 driver.c

Figure 23 - driver.c

3.1.11 main.c

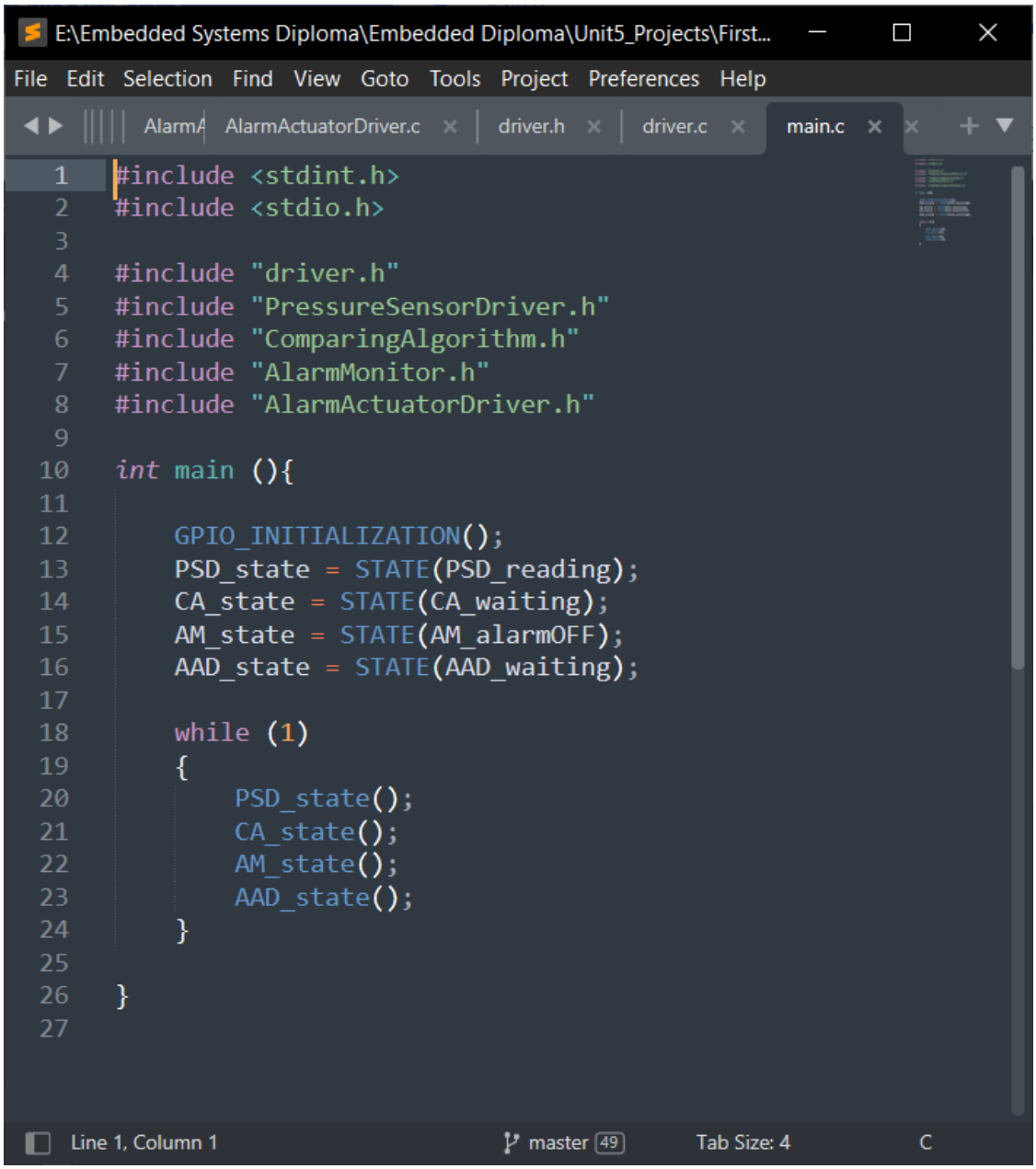


Figure 24 - main.c

3.1.12 startup.c

E:\Embedded Systems Diploma\Embedded Diploma\Unit5_Projects\First Term - Project ...

File Edit Selection Find View Goto Tools Project Preferences Help

dri driver.c main.c linker_script.ld startup.c ComparingAlgorithm.h

```c
int main(void);
void Reset_Handler();
void Default_Handler(){
    Reset_Handler();
}

void NMI_Handler()          __attribute__ ((weak, alias("Default_Handler")));
void H_Fault_Handler()      __attribute__ ((weak, alias("Default_Handler")));
void MM_Fault_Handler()     __attribute__ ((weak, alias("Default_Handler")));
void Bus_Fault()            __attribute__ ((weak, alias("Default_Handler")));
void Usage_Fault_Handler()  __attribute__ ((weak, alias("Default_Handler")));

extern uint32_t _STACK_top;

uint32_t vectors[] __attribute__((section(".vectors"))) = {
    (uint32_t) &_STACK_top,
    (uint32_t) &Reset_Handler,
    (uint32_t) &NMI_Handler,
    (uint32_t) &H_Fault_Handler,
    (uint32_t) &MM_Fault_Handler,
    (uint32_t) &Bus_Fault,
    (uint32_t) &Usage_Fault_Handler
};

extern unsigned int _E_text_;
extern unsigned int _S_data_;
extern unsigned int _E_data_;
extern unsigned int _S_bss_;
extern unsigned int _E_bss_;

void Reset_Handler(){
    //Copy .data from FLASH to SRAM
    unsigned int data_SIZE =
        (unsigned char*)(&_E_data_) - (unsigned char*)(&_S_data_);

    unsigned char * P_src = (unsigned char*)_E_text_;
    unsigned char * P_dest = (unsigned char*)_S_data_;

    for(int i = 0; i < data_SIZE; i++){
        *((unsigned char *)P_dest++) = *((unsigned char *)P_src++);
    }

    //Initialize .bss in SRAM by zeros
    unsigned int bss_SIZE =
        (unsigned char*)(&_E_bss_) - (unsigned char*)(&_S_bss_);

    P_dest = (unsigned char*)_S_bss_;

    for(int i = 0; i < bss_SIZE; i++){
        *((unsigned char *)P_dest++) = (unsigned char) 0;
    }

    //Proceed to main function
    main();
}
```

Line 1, Column 1 master 49 Tab Size: 4 C

Figure 25 - startup.c

3.1.13 linker_script.ld

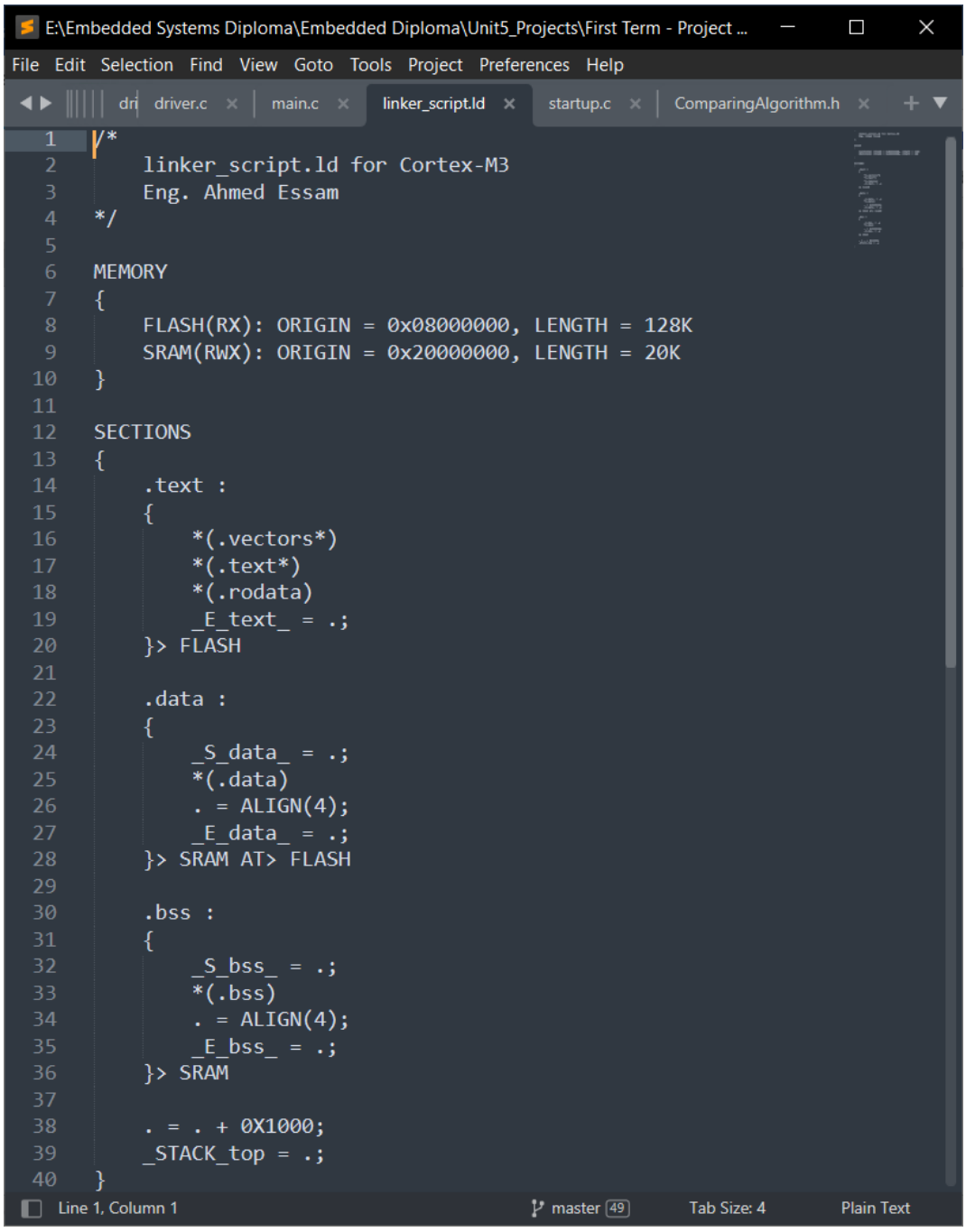


Figure 26 - linker_script.ld

3.1.14 Makefile

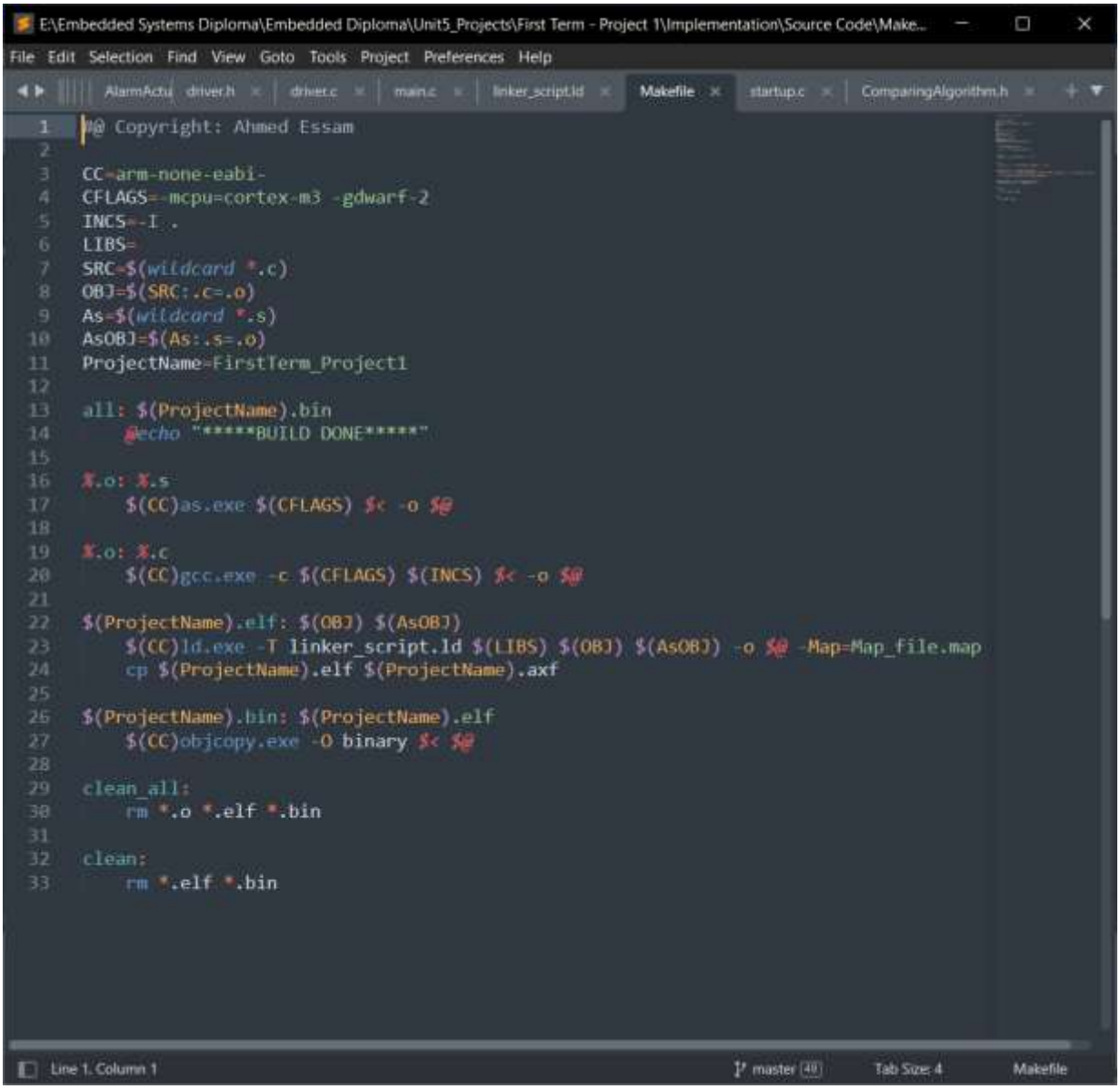


Figure 27 - Makefile

3.2 Output

3.2.1 Simulation Screenshots

Here, the pressure value was 50 which is higher than the threshold. So, the alarm was set to be turned on.

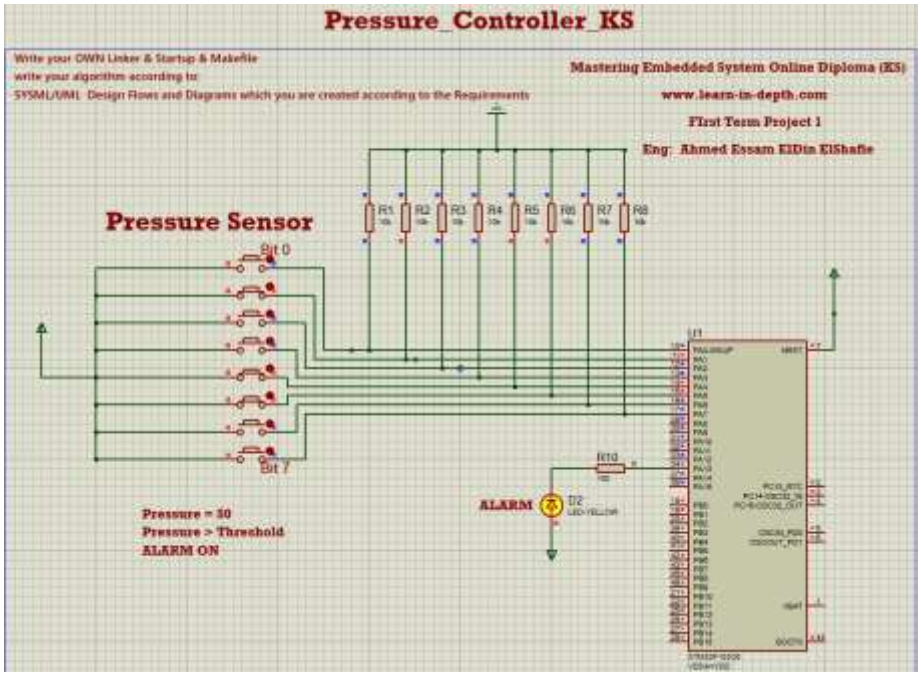


Figure 28 - Pressure = 50 Simulation

Here, the pressure value was 18 which is lower than the threshold. So, the alarm was off.

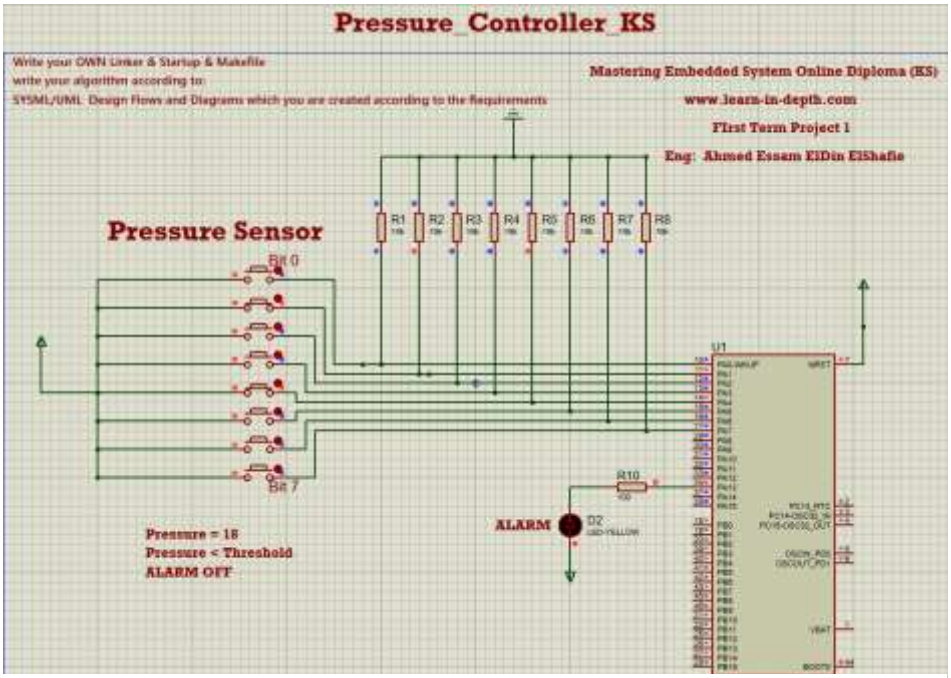


Figure 29 - Pressure = 18 Simulation

Here, the pressure value was 20 which is equal to the threshold. So, the alarm was off.

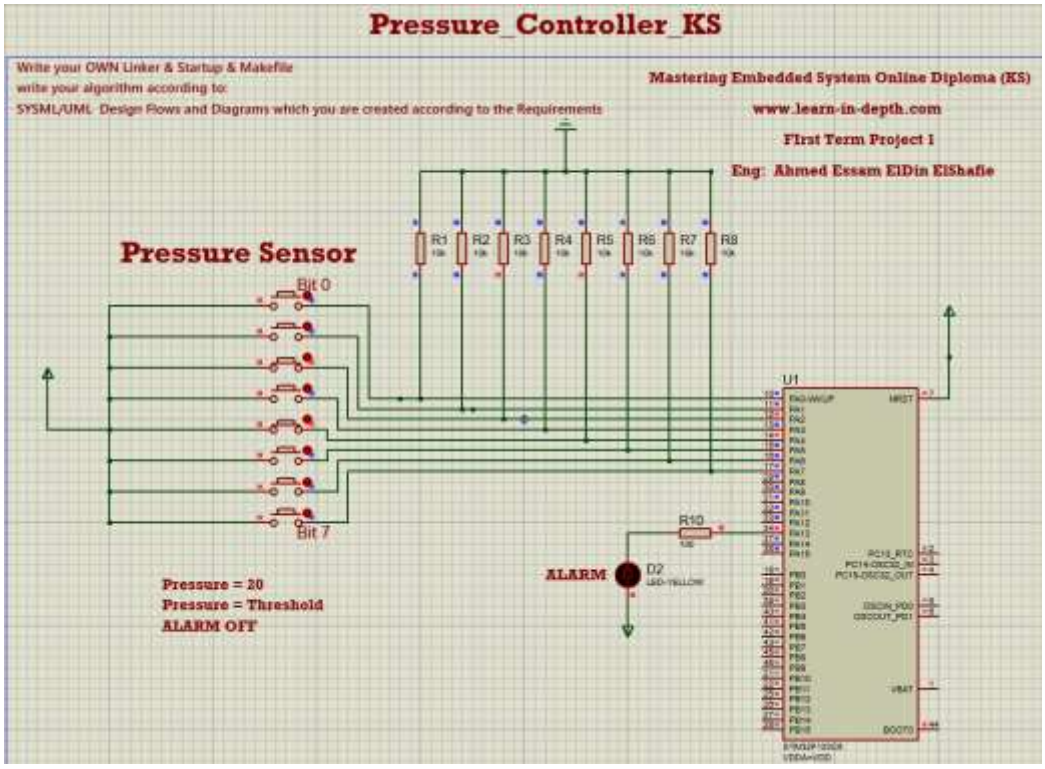


Figure 30 - Pressure = 18 Simulation

Here, the pressure value was 21 which is higher than the threshold. So, the alarm was on.

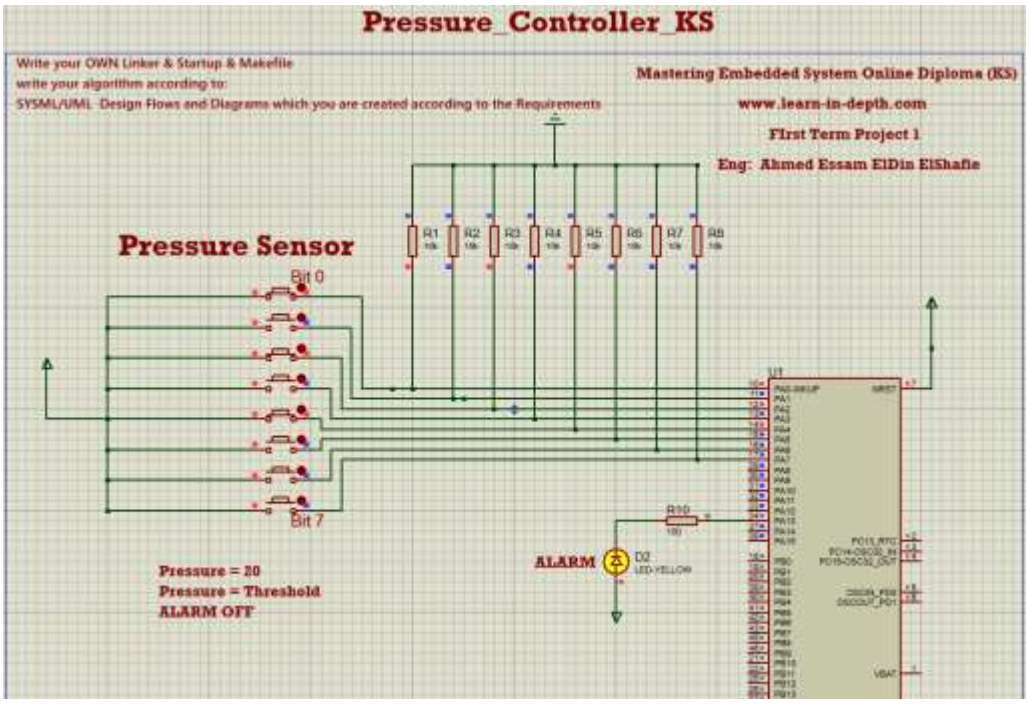
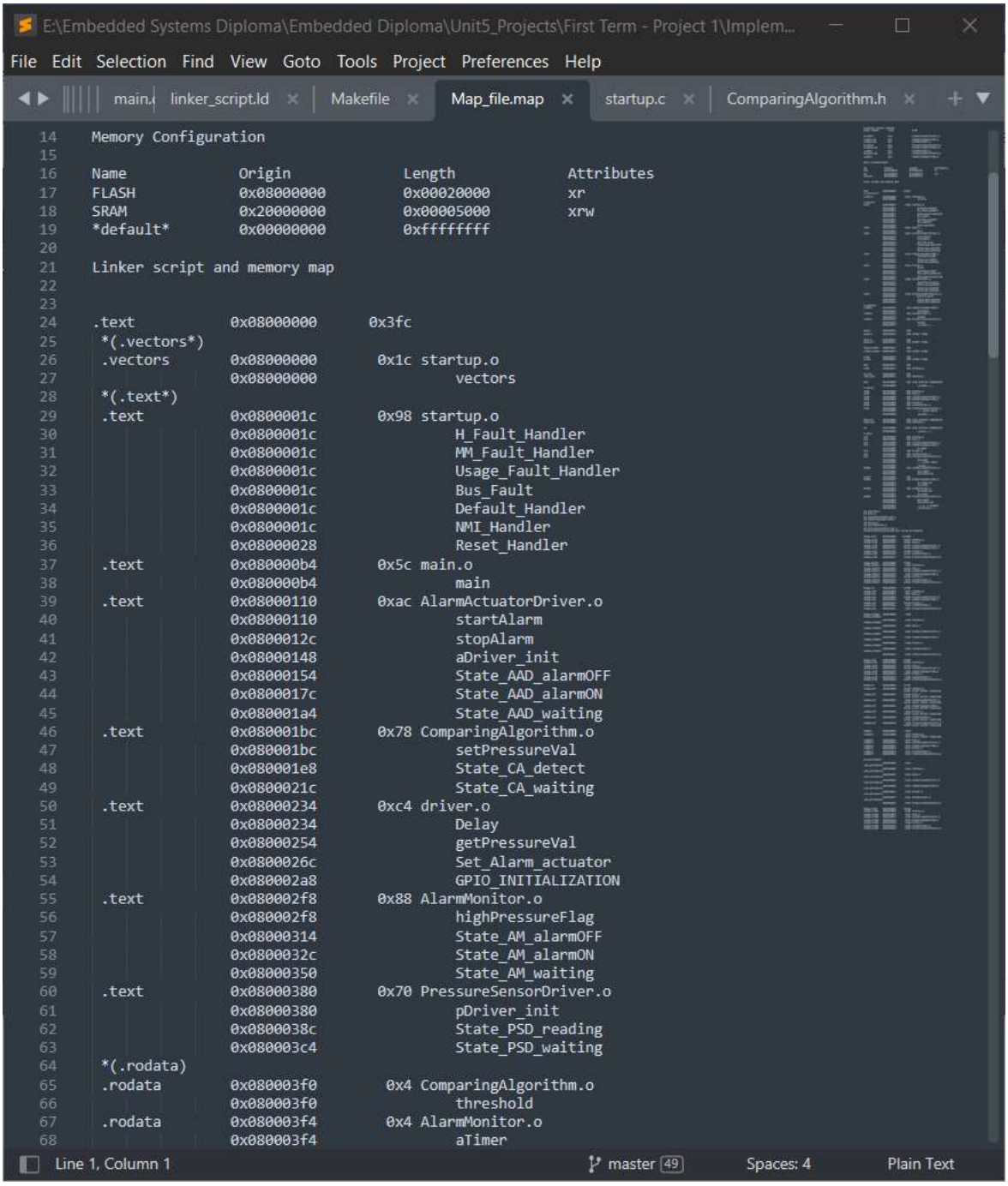


Figure 31 - Pressure = 21 Simulation

3.2.2 Output Mapfile




```
Memory Configuration

Name             Origin             Length             Attributes
FLASH            0x08000000         0x00020000         xr
SRAM             0x20000000         0x00005000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map


.text           0x08000000      0x3fc
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x0800001c       0x98 startup.o
                0x0800001c                H_Fault_Handler
                0x0800001c                MM_Fault_Handler
                0x0800001c                Usage_Fault_Handler
                0x0800001c                Bus_Fault
                0x0800001c                Default_Handler
                0x0800001c                NMI_Handler
                0x08000028                Reset_Handler
 .text          0x080000b4       0x5c main.o
                0x080000b4                main
 .text          0x08000110       0xac AlarmActuatorDriver.o
                0x08000110                startAlarm
                0x0800012c                stopAlarm
                0x08000148                aDriver_init
                0x08000154                State_AAD_alarmOFF
                0x0800017c                State_AAD_alarmON
                0x080001a4                State_AAD_waiting
 .text          0x080001bc       0x78 ComparingAlgorithm.o
                0x080001bc                setPressureVal
                0x080001e8                State_CA_detect
                0x0800021c                State_CA_waiting
 .text          0x08000234       0xc4 driver.o
                0x08000234                Delay
                0x08000254                getPressureVal
                0x0800026c                Set_Alarm_actuator
                0x080002a8                GPIO_INITIALIZATION
 .text          0x080002f8       0x88 AlarmMonitor.o
                0x080002f8                highPressureFlag
                0x08000314                State_AM_alarmOFF
                0x0800032c                State_AM_alarmON
                0x08000350                State_AM_waiting
 .text          0x08000380       0x70 PressureSensorDriver.o
                0x08000380                pDriver_init
                0x0800038c                State_PSD_reading
                0x080003c4                State_PSD_waiting
 *(.rodata)
 .rodata        0x080003f0        0x4 ComparingAlgorithm.o
                0x080003f0                threshold
 .rodata        0x080003f4        0x4 AlarmMonitor.o
                0x080003f4                aTimer
```

Figure 32 - Mapfile 1/2

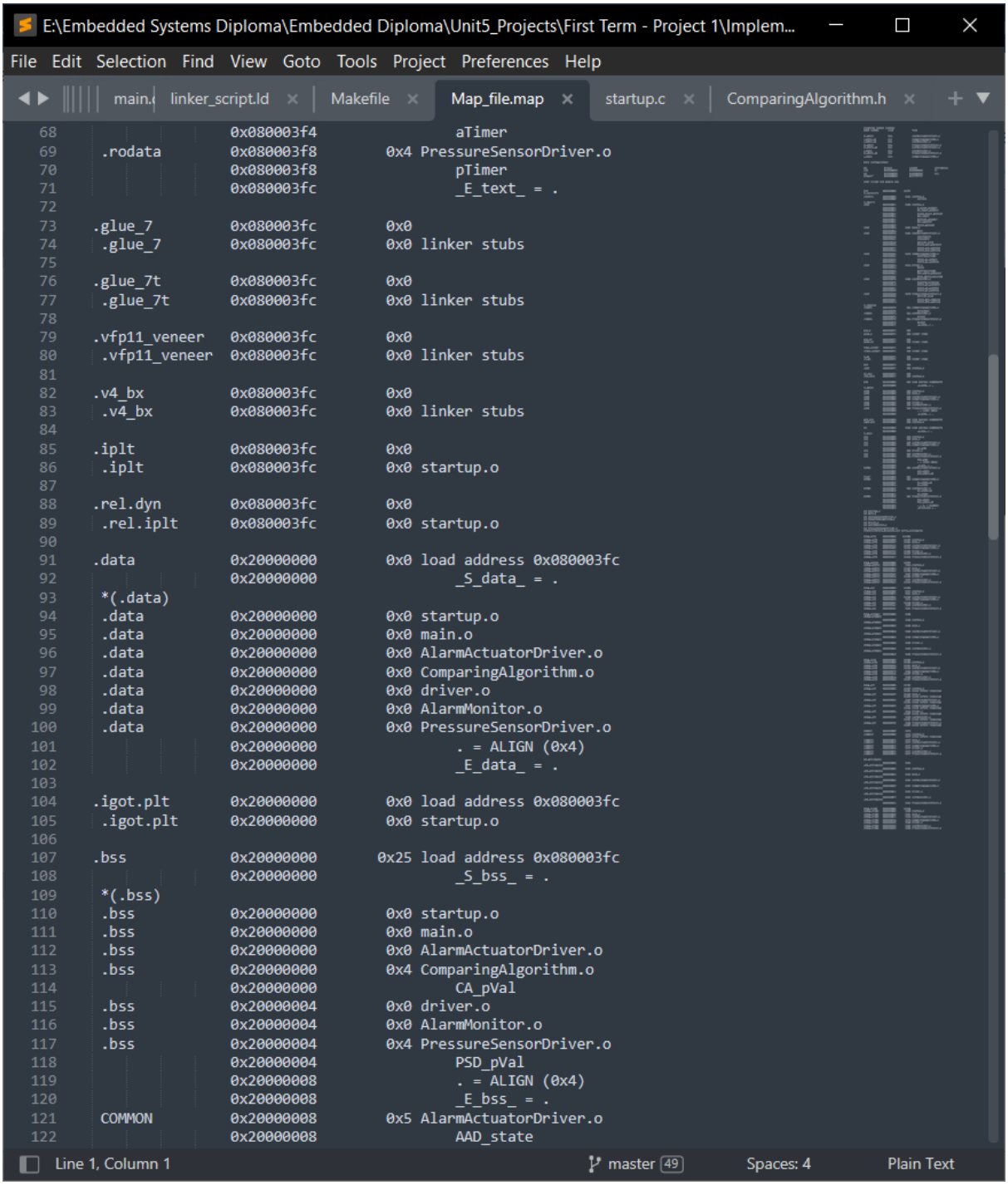


Figure 33 - Mapfile 2/2