



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

1 INTRODUCTION.....	3
2 DESIGN SEQUENCE	3
2.1 Case Study	3
2.2 SDLC Model	4
2.3 Requirement.....	5
2.4 System Exploration/Partitioning.....	6
2.5 System Analysis	7
2.5.1 Use Case Diagram	7
2.5.2 Activity Diagram	8
2.5.3 Sequence Diagram	8
2.6 System Design	9
2.6.1 Block Diagram	9
2.6.2 State Machines Diagram	10
2.6.3 State Machine Simulation	13
3 SOFTWARE IMPLEMENTATION.....	14
3.1 Source Code.....	14
3.1.1 PressureSensorDrive.h	14
3.1.2 PressureSensorDrive.c	15
3.1.3 ComparingAlgorithm.h.....	16
3.1.4 ComparingAlgorithm.c	17
3.1.5 AlarmMonitor.h.....	18
3.1.6 AlarmMonitor.c	19
3.1.7 AlarmAcutatorDriver.h	20
3.1.8 AlarmAcutatorDriver.c.....	21
3.1.9 driver.h	22
3.1.10 driver.c.....	23
3.1.11 main.c.....	24
3.1.12 startup.c	25
3.1.13 linker_script.ld	26
3.1.14 Makefile	27
3.2 Output	28
3.2.1 Simulation Screenshots.....	28
3.2.2 Output Mapfile.....	30

Table of Figures

Figure 1 - SDLC V-Model.....	4
Figure 2 - Requirement Diagram	5
Figure 3 - STM32F103C6	6
Figure 4 - STM32F103C6 Specifications	6
Figure 5 - Use Case Diagram	7
Figure 6 - Activity Diagram	8
Figure 7 - Sequence Diagram	8
Figure 8 - Block Diagram	9
Figure 9 - PressureSensorDriver State Machine.....	10
Figure 10 - ComparingAlgorithm State Machine	11
Figure 11 - AlarmMonitor State Machine	11
Figure 12 - AlarmActuatorDriver State Machine.....	12
Figure 13 - State Machine Simulation	13
Figure 14 - PressureSensorDrive.h	14
Figure 15 - PressureSensorDrive.c	15
Figure 16 - ComparingAlgorithm.h.....	16
Figure 17 - ComparingAlgorithm.c.....	17
Figure 18 - AlarmMonitor.h.....	18
Figure 19 - AlarmMonitor.c	19
Figure 20 - AlarmAcutatorDriver.h	20
Figure 21 - AlarmAcutatorDriver.c.....	21
Figure 22 - driver.h	22
Figure 23 - driver.c.....	23
Figure 24 - main.c.....	24
Figure 25 - startup.c	25
Figure 26 - linker_script.ld	26
Figure 27 - Makefile	27
Figure 28 - Pressure = 50 Simulation.....	28
Figure 29 - Pressure = 18 Simulation.....	28
Figure 30 - Pressure = 18 Simulation.....	29
Figure 31 - Pressure = 21 Simulation.....	29
Figure 32 - Mapfile 1/2.....	30
Figure 33 - Mapfile 2/2.....	31

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.

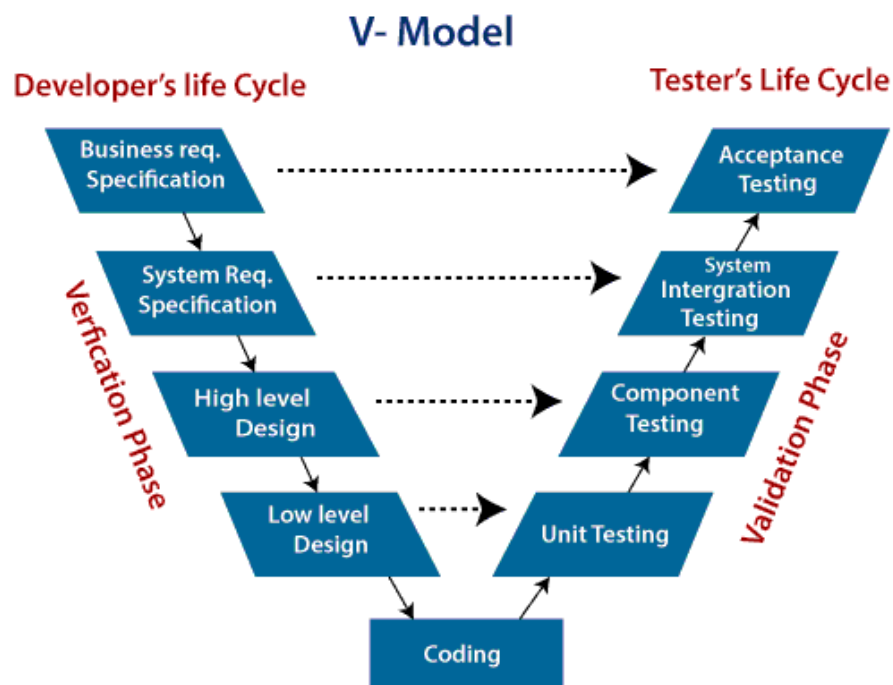


Figure 1 - SDLC V-Model

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.

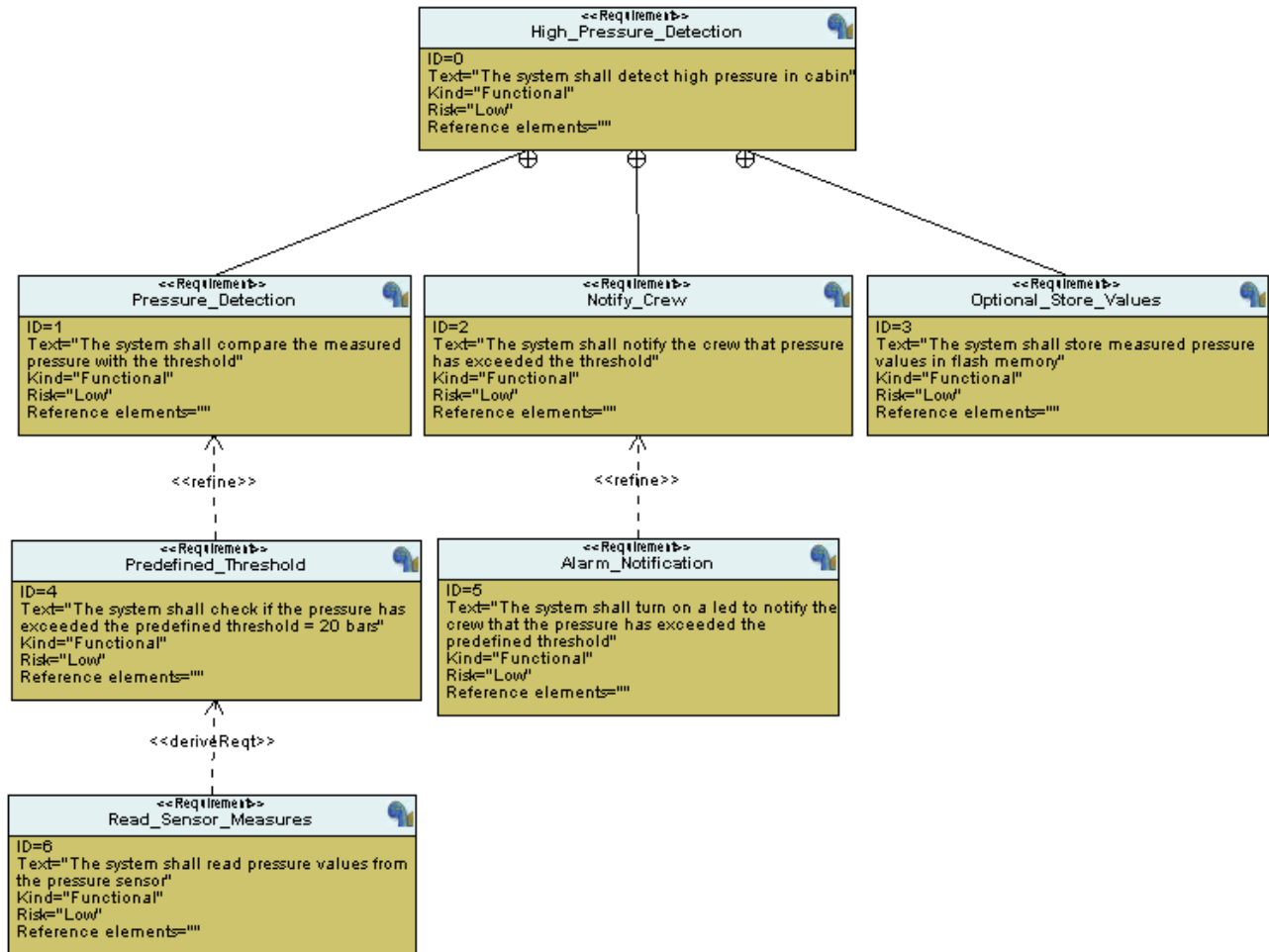


Figure 2 - Requirement Diagram

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.

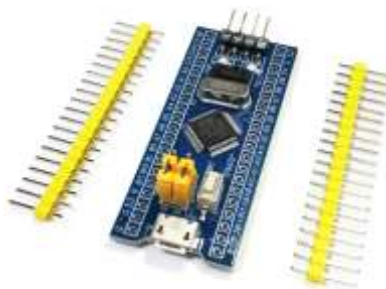


Figure 3 - STM32F103C6

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

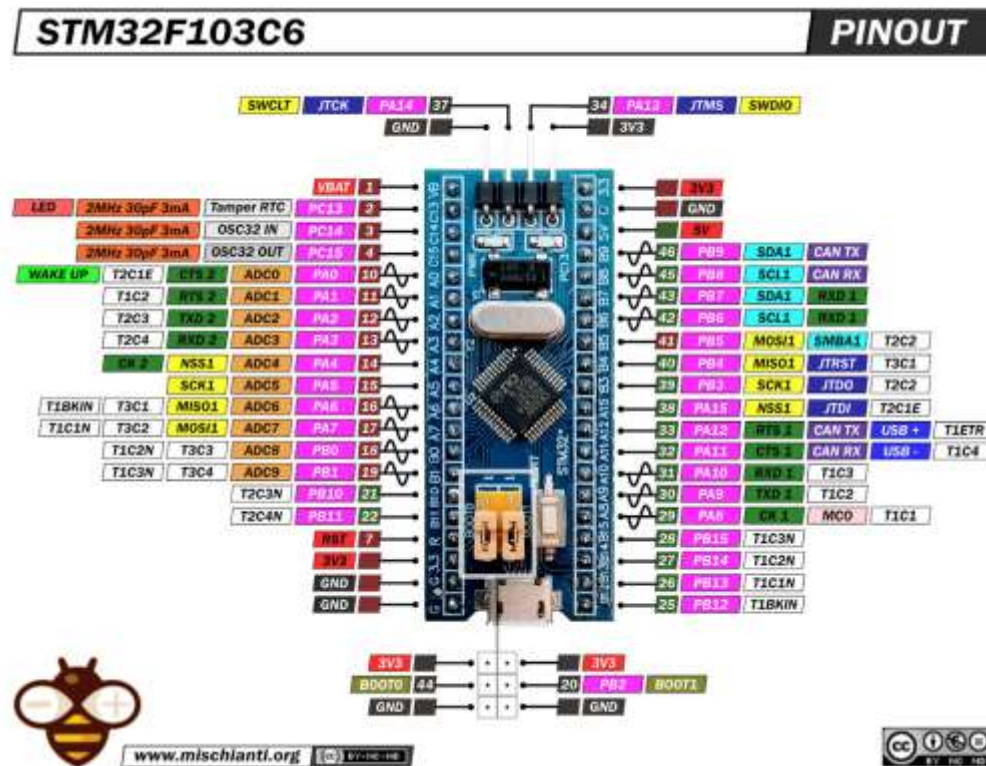


Figure 4 - STM32F103C6 Specifications

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.

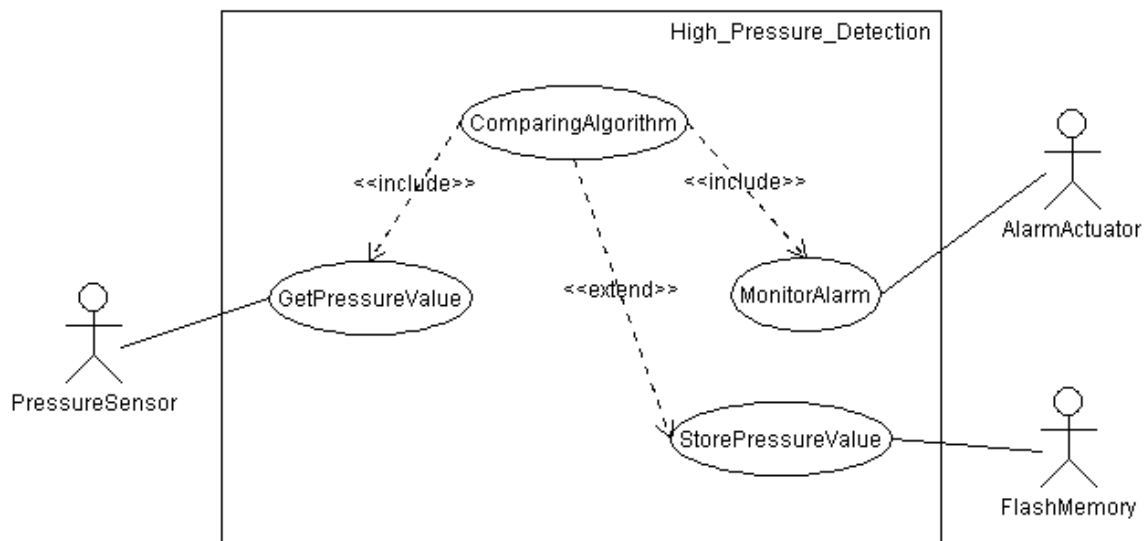


Figure 5 - Use Case Diagram

2.5.2 Activity Diagram

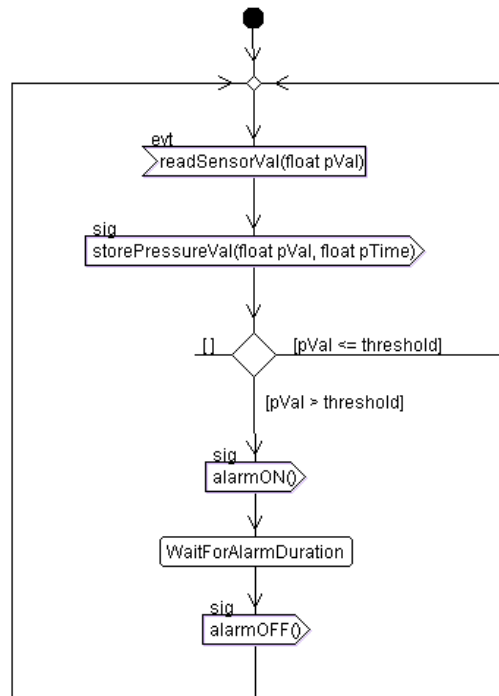


Figure 6 - Activity Diagram

2.5.3 Sequence Diagram

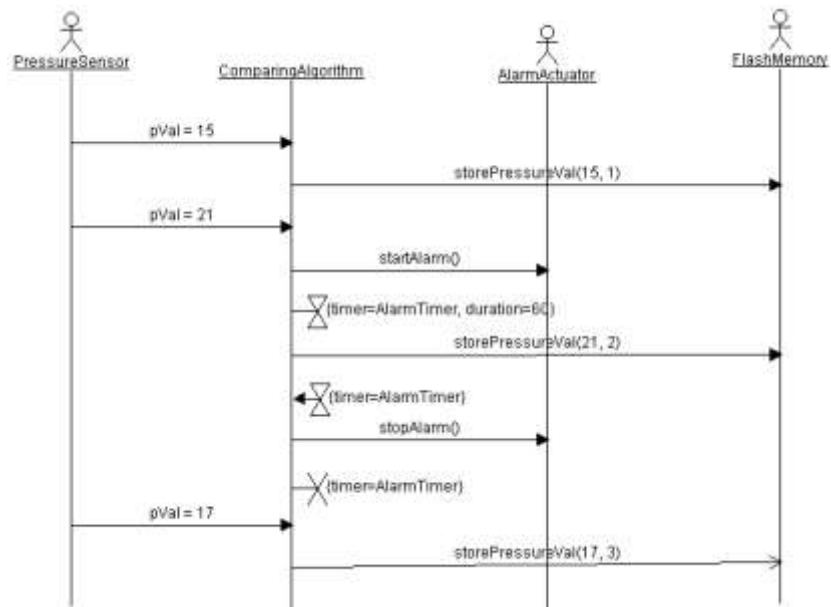


Figure 7 - Sequence Diagram

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().

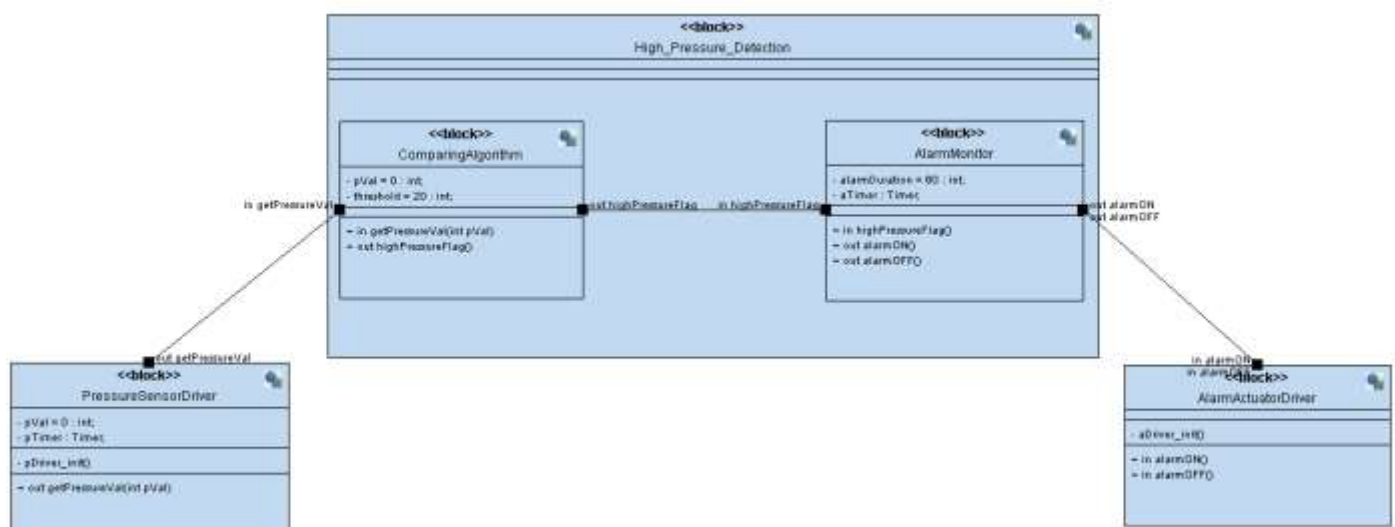


Figure 8 - Block Diagram

2.6.2 State Machines Diagram

2.6.2.1 PressureSensorDriver State Machine

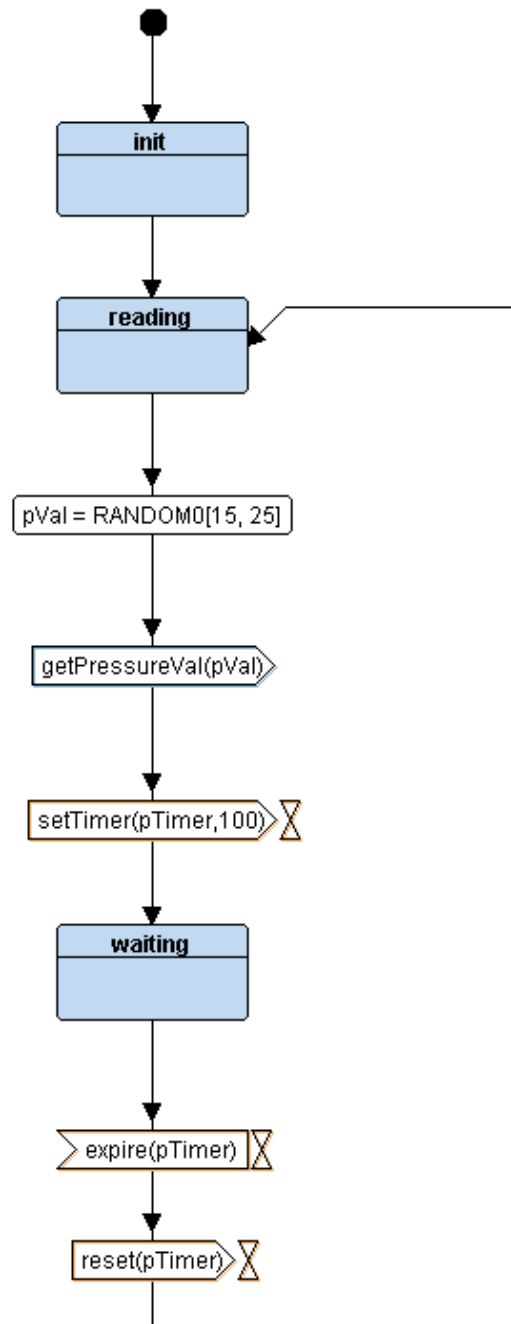


Figure 9 - PressureSensorDriver State Machine

2.6.2.2 ComparingAlgorithm State Machine

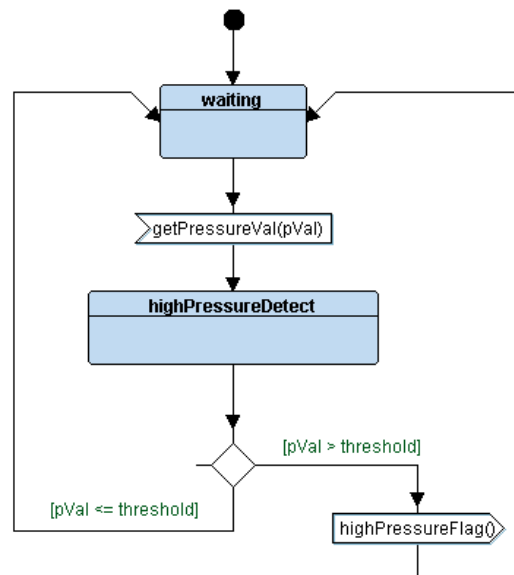


Figure 10 - ComparingAlgorithm State Machine

2.6.2.3 AlarmMonitor State Machine

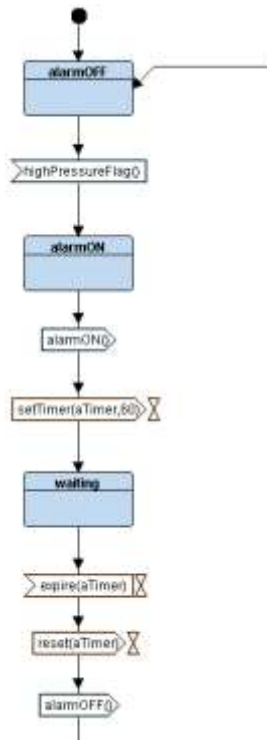


Figure 11 - AlarmMonitor State Machine

2.6.2.4 AlarmActuatorDriver State Machine

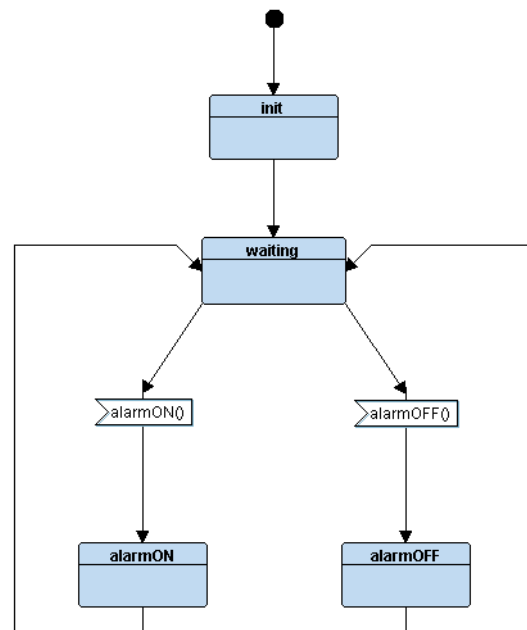


Figure 12 - AlarmActuatorDriver State Machine

2.6.3 State Machine Simulation

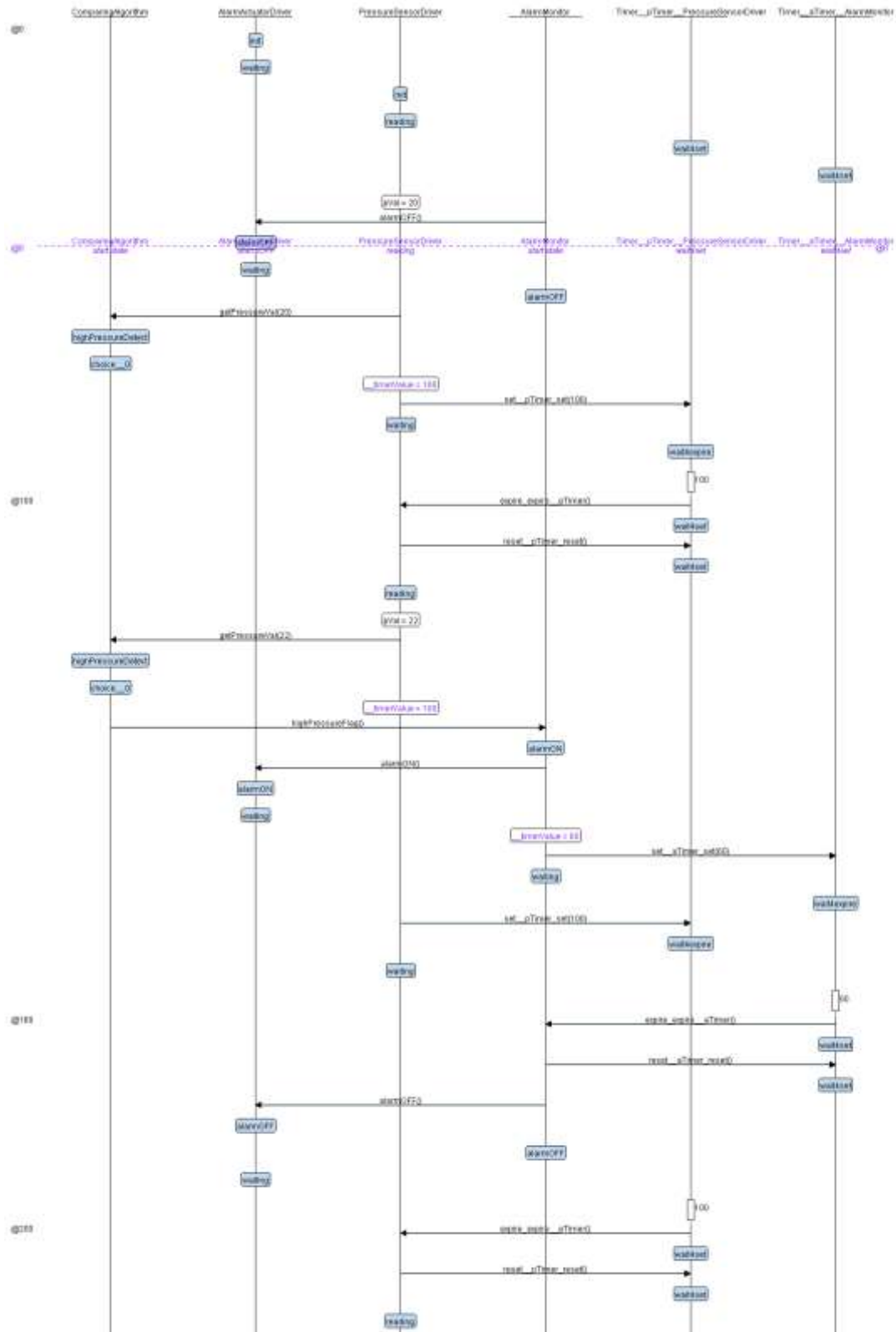


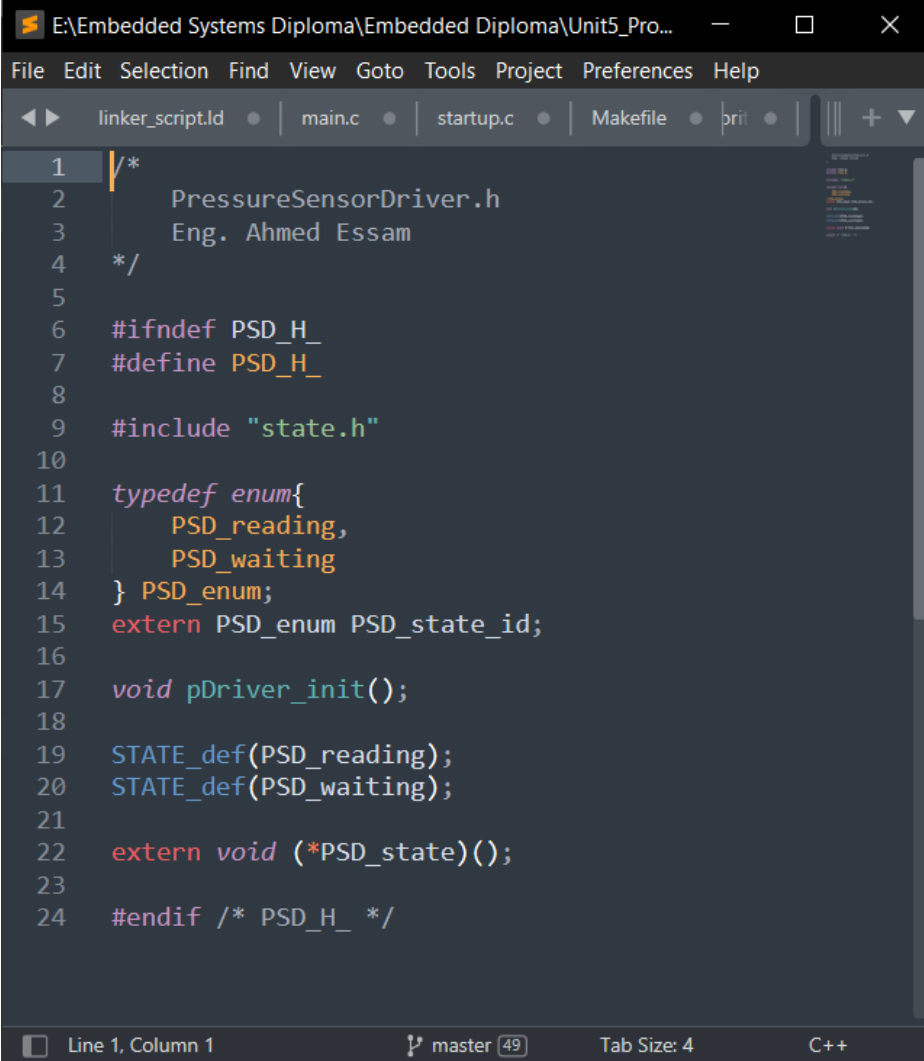
Figure 13 - State Machine Simulation

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



```
1  /*
2      PressureSensorDriver.h
3      Eng. Ahmed Essam
4  */
5
6  #ifndef PSD_H_
7  #define PSD_H_
8
9  #include "state.h"
10
11  typedef enum{
12      PSD_reading,
13      PSD_waiting
14  } PSD_enum;
15  extern PSD_enum PSD_state_id;
16
17  void pDriver_init();
18
19  STATE_def(PSD_reading);
20  STATE_def(PSD_waiting);
21
22  extern void (*PSD_state)();
23
24  #endif /* PSD_H_ */
```

Figure 14 - PressureSensorDrive.h

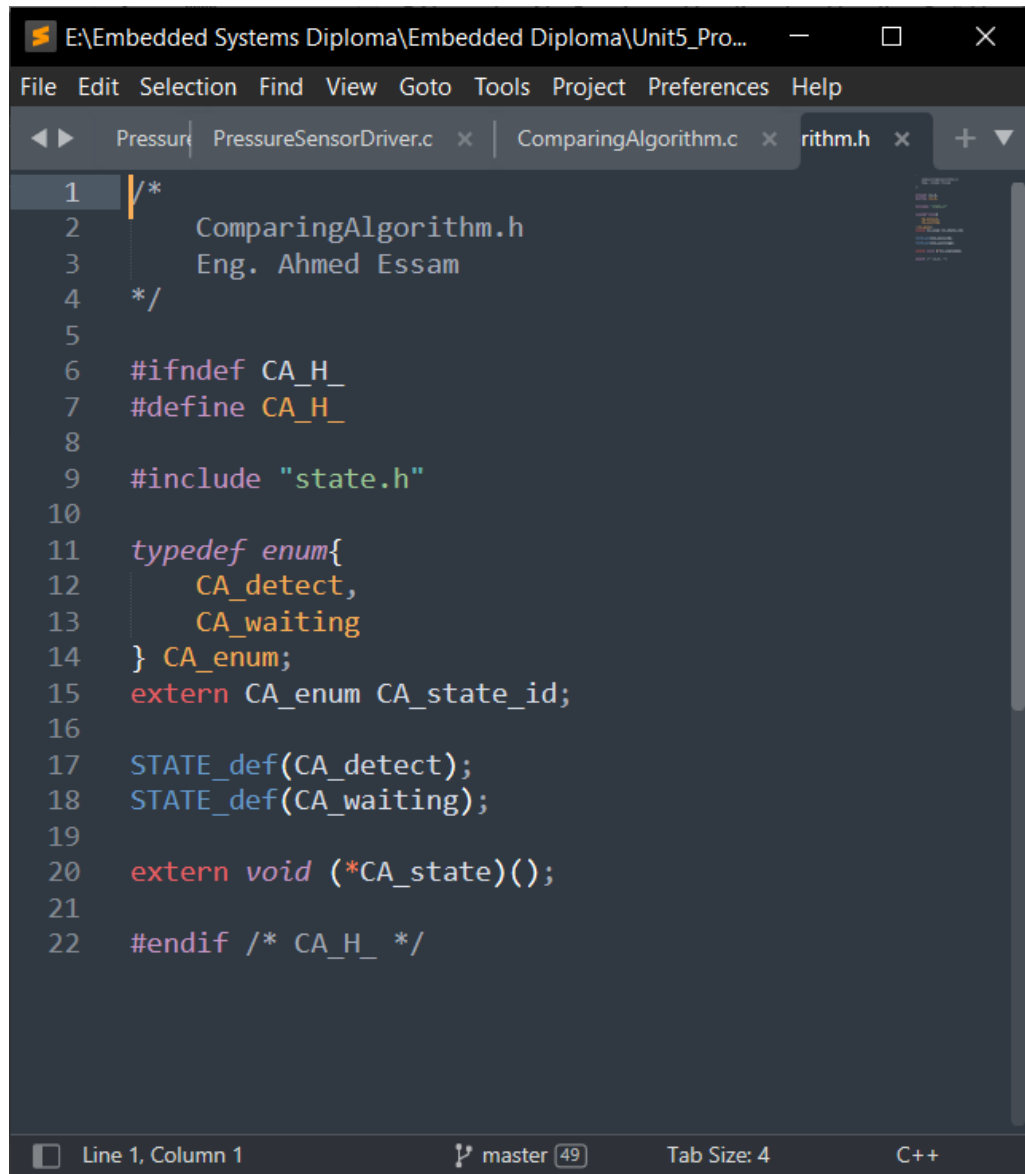
3.1.2 PressureSensorDrive.c



```
1  /*
2   PressureSensorDriver.c
3   Eng. Ahmed Essam
4  */
5
6  #include "PressureSensorDriver.h"
7  #include "driver.h"
8
9  int PSD_pVal = 0;
10 const int pTimer = 100000;
11
12 PSD_enum PSD_state_id;
13 void (*PSD_state)();
14
15 void pDriver_init(){
16     //Initialize PressureSensorDriver
17 }
18
19 STATE_def(PSD_reading){
20
21     //State Name
22     PSD_state_id = PSD_reading;
23
24     //State Action
25     PSD_pVal = getPressureVal();
26     setPressureVal(PSD_pVal);
27     PSD_state = STATE(PSD_waiting);
28 }
29
30 STATE_def(PSD_waiting){
31
32     //State Name
33     PSD_state_id = PSD_waiting;
34
35     //State Action
36     Delay(pTimer);
37     PSD_state = STATE(PSD_reading);
38 }
```

Figure 15 - PressureSensorDrive.c

3.1.3 ComparingAlgorithm.h

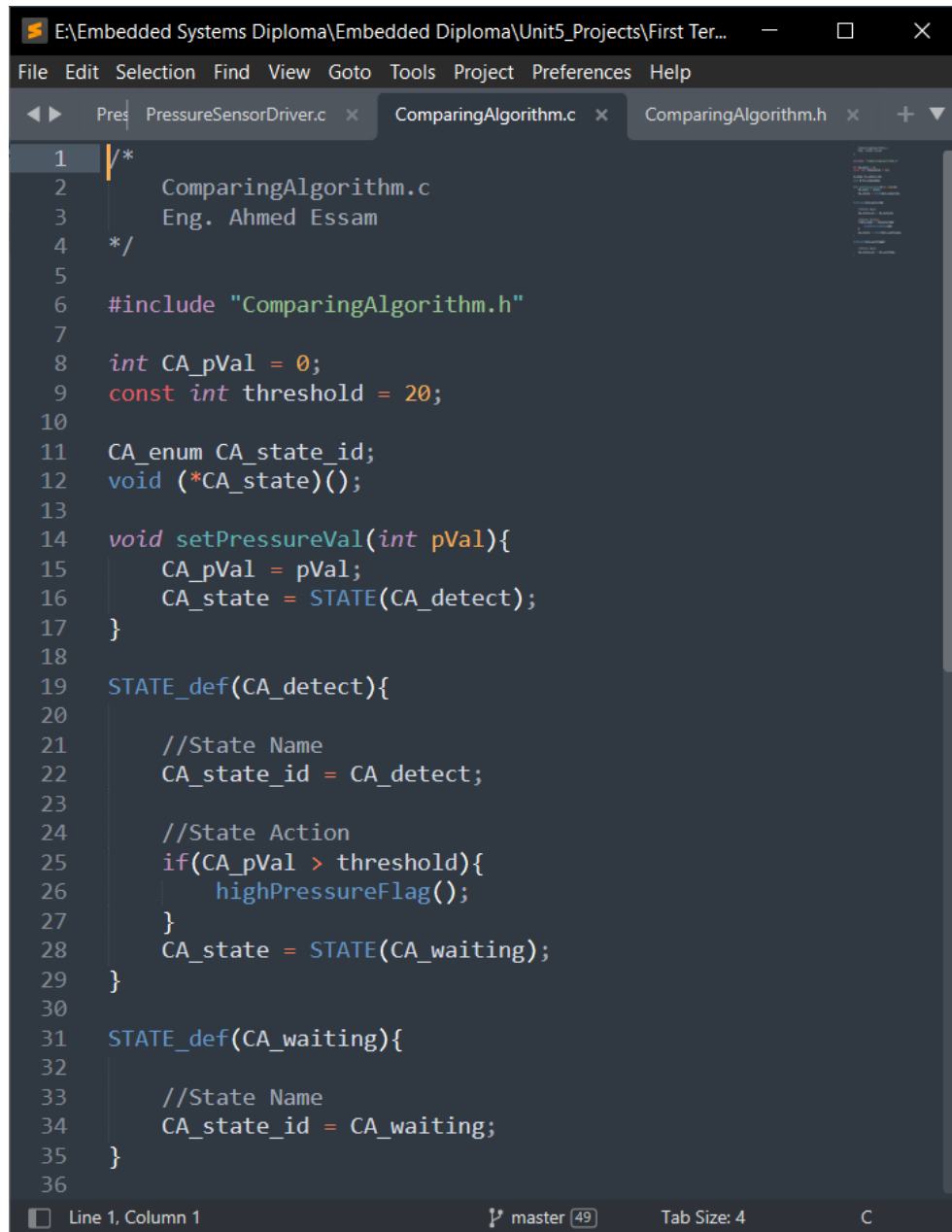


```
1  /*
2      ComparingAlgorithm.h
3      Eng. Ahmed Essam
4  */
5
6  #ifndef CA_H_
7  #define CA_H_
8
9  #include "state.h"
10
11  typedef enum{
12      CA_detect,
13      CA_waiting
14  } CA_enum;
15  extern CA_enum CA_state_id;
16
17  STATE_def(CA_detect);
18  STATE_def(CA_waiting);
19
20  extern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

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

Figure 16 - ComparingAlgorithm.h

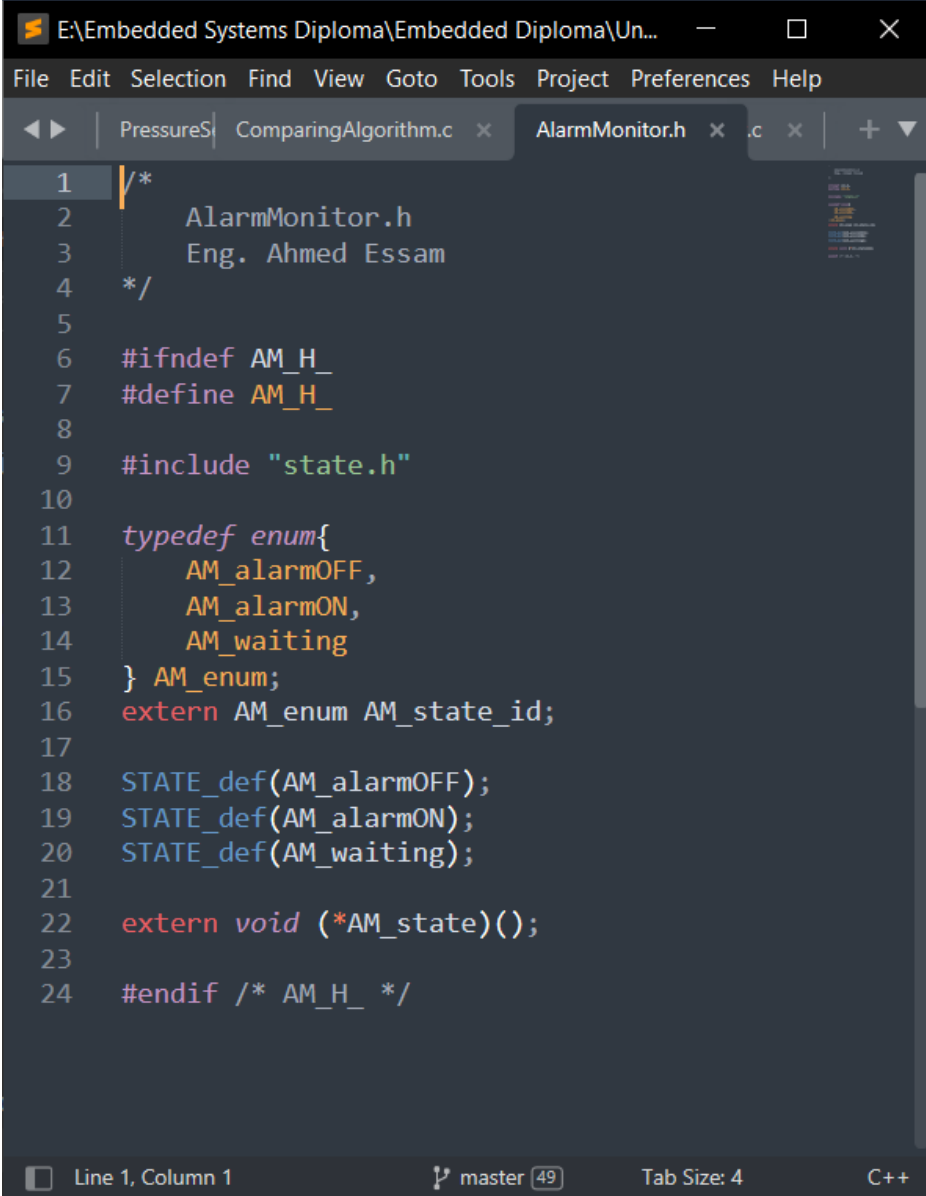
3.1.4 ComparingAlgorithm.c



```
1  /*
2     ComparingAlgorithm.c
3     Eng. Ahmed Essam
4  */
5
6  #include "ComparingAlgorithm.h"
7
8  int CA_pVal = 0;
9  const int threshold = 20;
10
11  CA_enum CA_state_id;
12  void (*CA_state)();
13
14  void setPressureVal(int pVal){
15      CA_pVal = pVal;
16      CA_state = STATE(CA_detect);
17  }
18
19  STATE_def(CA_detect){
20
21      //State Name
22      CA_state_id = CA_detect;
23
24      //State Action
25      if(CA_pVal > threshold){
26          highPressureFlag();
27      }
28      CA_state = STATE(CA_waiting);
29  }
30
31  STATE_def(CA_waiting){
32
33      //State Name
34      CA_state_id = CA_waiting;
35  }
36
```

Figure 17 - ComparingAlgorithm.c

3.1.5 AlarmMonitor.h



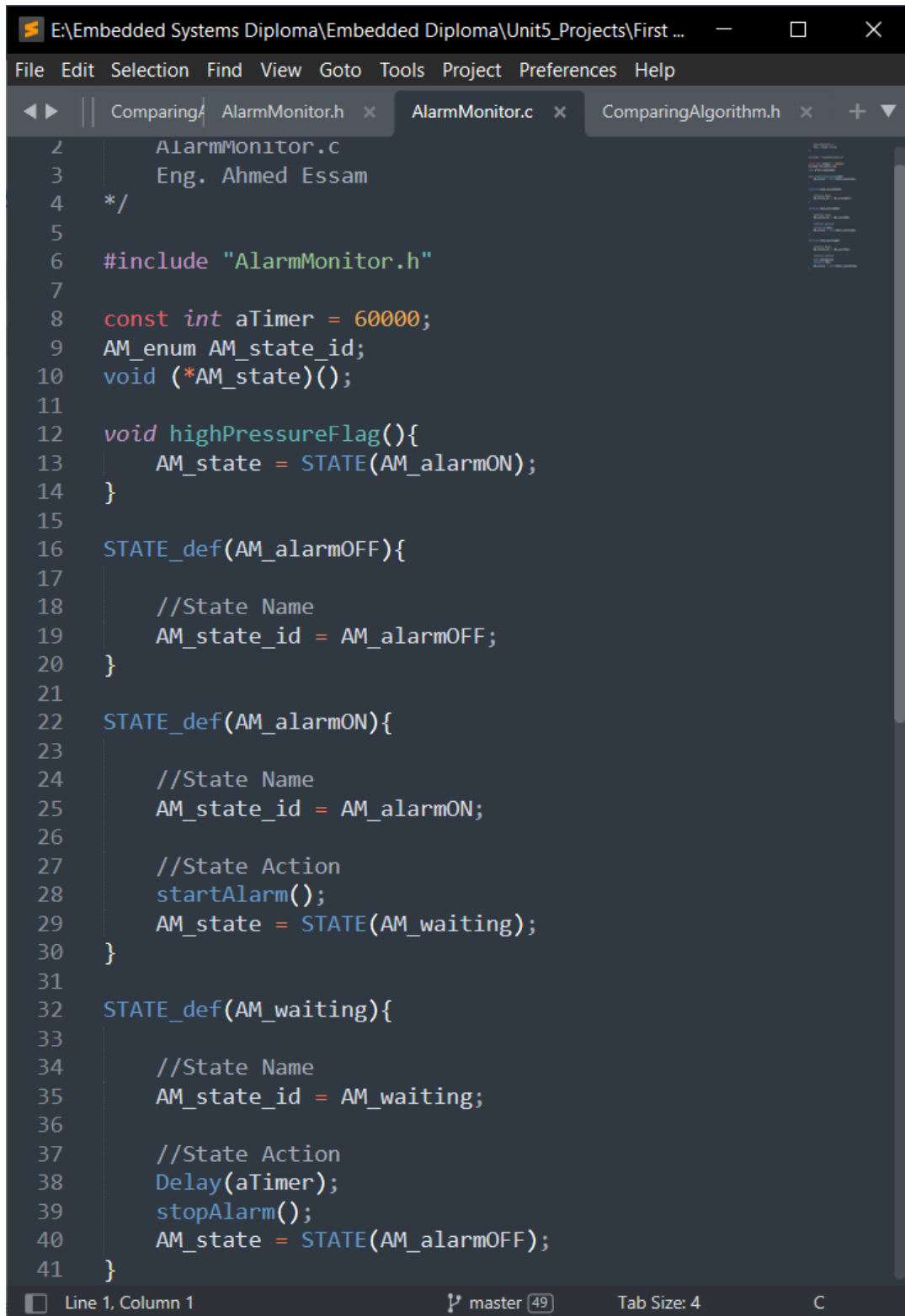
The screenshot shows a code editor window with the title bar "E:\Embedded Systems Diploma\Embedded Diploma\Un...". The menu bar includes "File", "Edit", "Selection", "Find", "View", "Goto", "Tools", "Project", "Preferences", and "Help". The tab bar shows three tabs: "PressureS", "ComparingAlgorithm.c", and "AlarmMonitor.h". The "AlarmMonitor.h" tab is active, displaying the following C++ code:

```
1  /*
2      AlarmMonitor.h
3      Eng. Ahmed Essam
4  */
5
6  #ifndef AM_H_
7  #define AM_H_
8
9  #include "state.h"
10
11  typedef enum{
12      AM_alarmOFF,
13      AM_alarmON,
14      AM_waiting
15  } AM_enum;
16  extern AM_enum AM_state_id;
17
18  STATE_def(AM_alarmOFF);
19  STATE_def(AM_alarmON);
20  STATE_def(AM_waiting);
21
22  extern void (*AM_state)();
23
24  #endif /* AM_H_ */
```

The status bar at the bottom shows "Line 1, Column 1", a branch indicator with "master (49)", "Tab Size: 4", and "C++".

Figure 18 - AlarmMonitor.h

3.1.6 AlarmMonitor.c

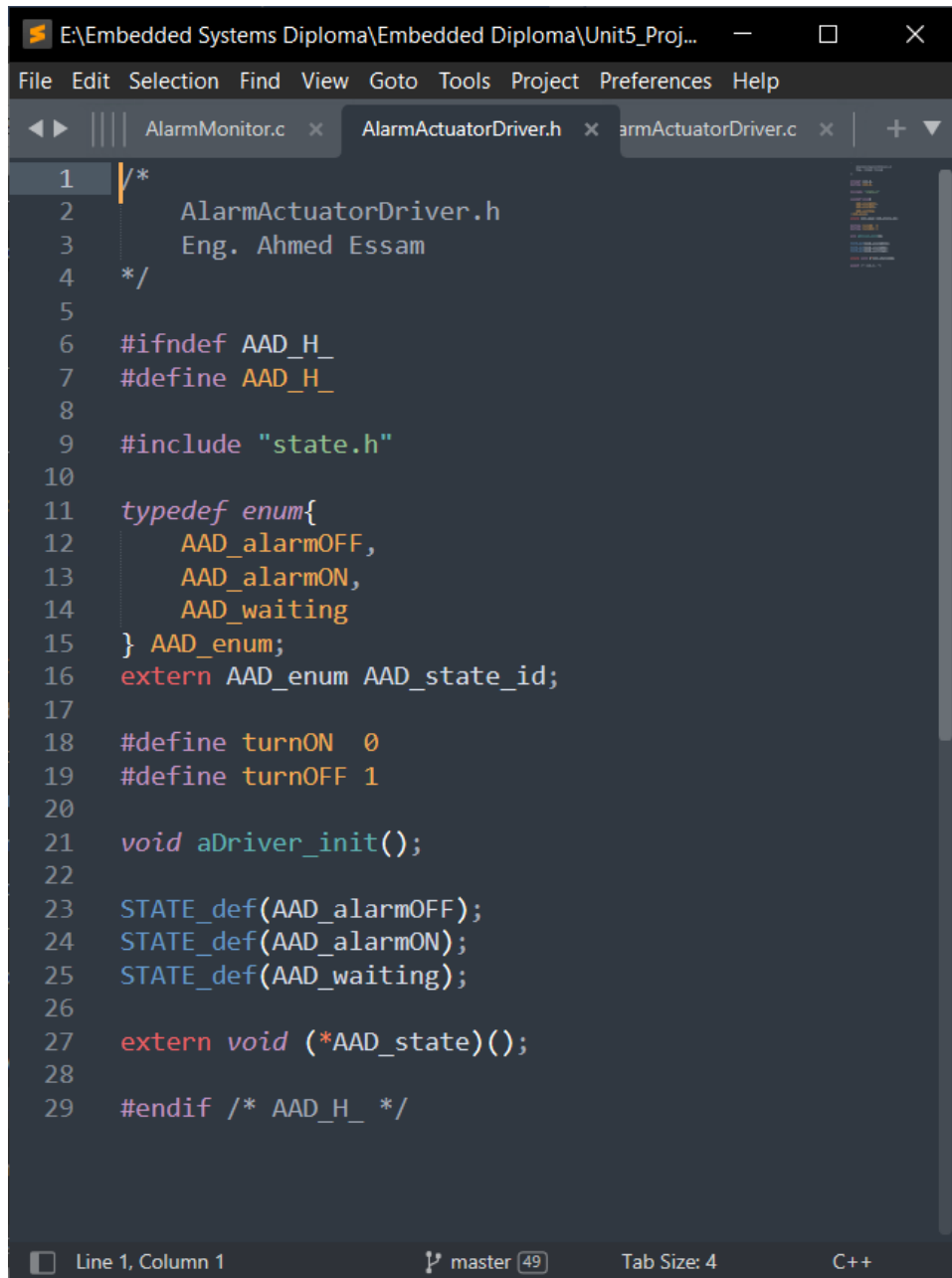


```
2      AlarmMonitor.c
3      Eng. Ahmed Essam
4      */
5
6      #include "AlarmMonitor.h"
7
8      const int aTimer = 60000;
9      AM_enum AM_state_id;
10     void (*AM_state)();
11
12     void highPressureFlag(){
13         AM_state = STATE(AM_alarmON);
14     }
15
16     STATE_def(AM_alarmOFF){
17
18         //State Name
19         AM_state_id = AM_alarmOFF;
20     }
21
22     STATE_def(AM_alarmON){
23
24         //State Name
25         AM_state_id = AM_alarmON;
26
27         //State Action
28         startAlarm();
29         AM_state = STATE(AM_waiting);
30     }
31
32     STATE_def(AM_waiting){
33
34         //State Name
35         AM_state_id = AM_waiting;
36
37         //State Action
38         Delay(aTimer);
39         stopAlarm();
40         AM_state = STATE(AM_alarmOFF);
41     }
```

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

Figure 19 - AlarmMonitor.c

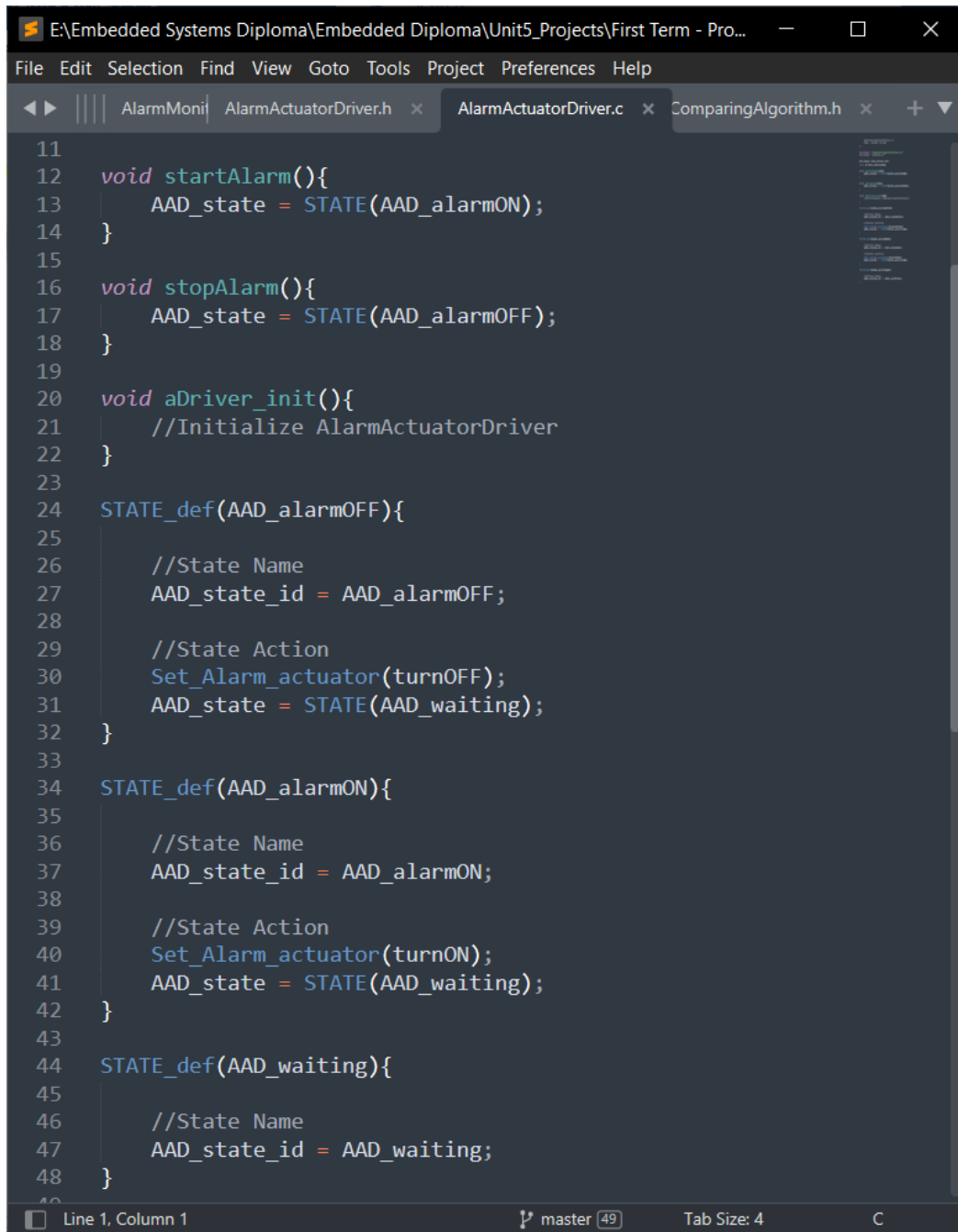
3.1.7 AlarmAcuatorDriver.h



```
1  /*
2      AlarmAcuatorDriver.h
3      Eng. Ahmed Essam
4  */
5
6  #ifndef AAD_H_
7  #define AAD_H_
8
9  #include "state.h"
10
11  typedef enum{
12      AAD_alarmOFF,
13      AAD_alarmON,
14      AAD_waiting
15  } AAD_enum;
16  extern AAD_enum AAD_state_id;
17
18  #define turnON 0
19  #define turnOFF 1
20
21  void aDriver_init();
22
23  STATE_def(AAD_alarmOFF);
24  STATE_def(AAD_alarmON);
25  STATE_def(AAD_waiting);
26
27  extern void (*AAD_state)();
28
29  #endif /* AAD_H_ */
```

Figure 20 - AlarmAcuatorDriver.h

3.1.8 AlarmAcutatorDriver.c



```
11
12 void startAlarm(){
13     AAD_state = STATE(AAD_alarmON);
14 }
15
16 void stopAlarm(){
17     AAD_state = STATE(AAD_alarmOFF);
18 }
19
20 void aDriver_init(){
21     //Initialize AlarmActuatorDriver
22 }
23
24 STATE_def(AAD_alarmOFF){
25
26     //State Name
27     AAD_state_id = AAD_alarmOFF;
28
29     //State Action
30     Set_Alarm_actuator(turnOFF);
31     AAD_state = STATE(AAD_waiting);
32 }
33
34 STATE_def(AAD_alarmON){
35
36     //State Name
37     AAD_state_id = AAD_alarmON;
38
39     //State Action
40     Set_Alarm_actuator(turnON);
41     AAD_state = STATE(AAD_waiting);
42 }
43
44 STATE_def(AAD_waiting){
45
46     //State Name
47     AAD_state_id = AAD_waiting;
48 }
```

Figure 21 - AlarmAcutatorDriver.c

3.1.9 driver.h



```
1 #include <stdint.h>
2 #include <stdio.h>
3
4 #define SET_BIT(ADDRESS,BIT) ADDRESS |= (1<<BIT)
5 #define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
6 #define TOGGLE_BIT(ADDRESS,BIT) ADDRESS ^= (1<<BIT)
7 #define READ_BIT(ADDRESS,BIT) ((ADDRESS) & (1<<(BIT)))
8
9
10 #define GPIO_PORTA 0x40010800
11 #define BASE_RCC 0x40021000
12
13 #define APB2ENR *(volatile uint32_t *) (BASE_RCC + 0x18)
14
15 #define GPIOA_CRL *(volatile uint32_t *) (GPIO_PORTA + 0x00)
16 #define GPIOA_CRH *(volatile uint32_t *) (GPIO_PORTA + 0x04)
17 #define GPIOA_IDR *(volatile uint32_t *) (GPIO_PORTA + 0x08)
18 #define GPIOA_ODR *(volatile uint32_t *) (GPIO_PORTA + 0x0C)
19
20
21 void Delay(int nCount);
22 int getPressureVal();
23 void Set_Alarm_actuator(int i);
24 void GPIO_INITIALIZATION ();
25
```

Figure 22 - driver.h

3.1.10 driver.c

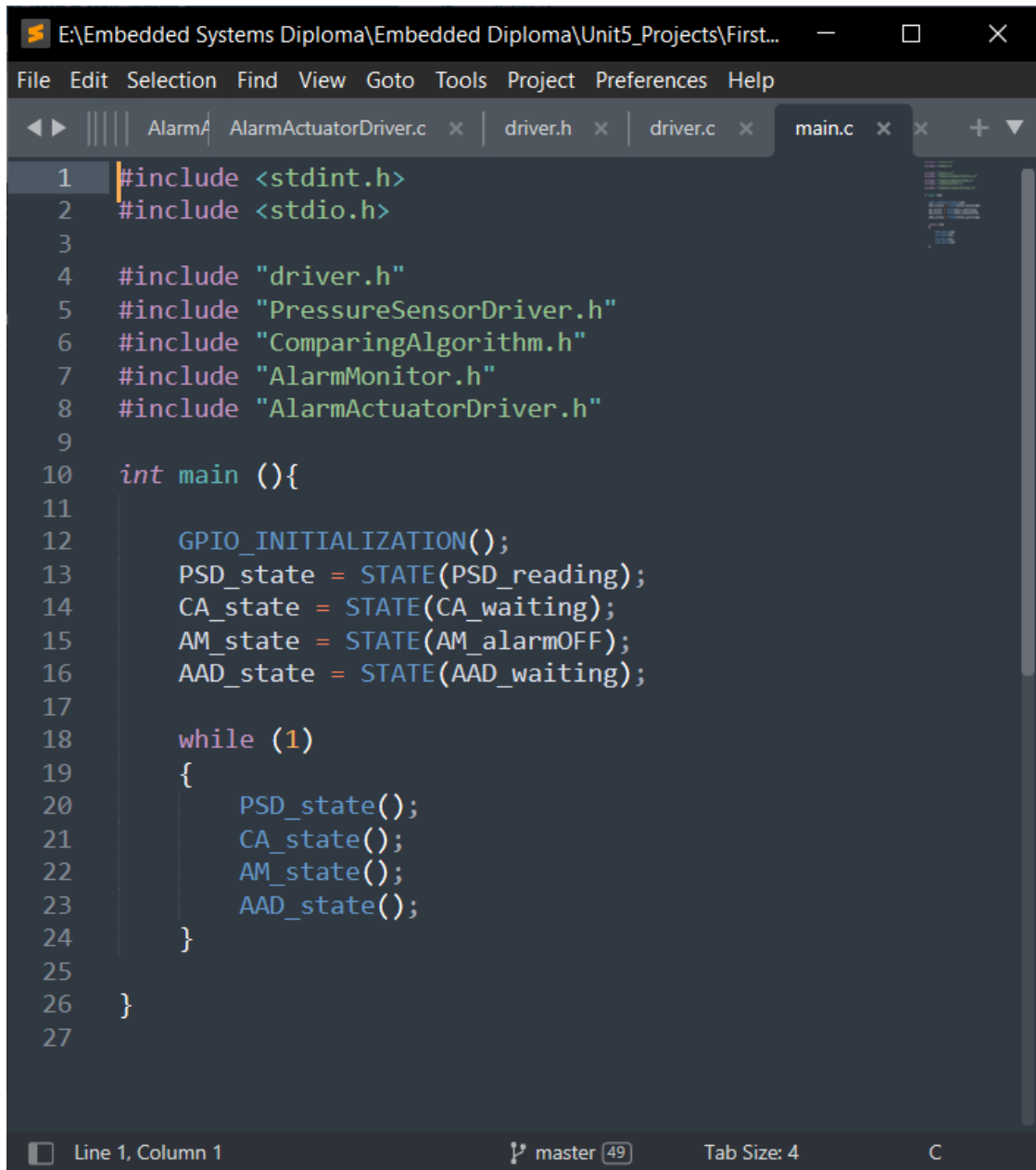


```
1 #include "driver.h"
2 #include <stdint.h>
3 #include <stdio.h>
4 void Delay(int nCount)
5 {
6     for(; nCount != 0; nCount--);
7 }
8
9 int getPressureVal(){
10     return (GPIOA_IDR & 0xFF);
11 }
12
13 void Set_Alarm_actuator(int i){
14     if (i == 1){
15         SET_BIT(GPIOA_ODR,13);
16     }
17     else if (i == 0){
18         RESET_BIT(GPIOA_ODR,13);
19     }
20 }
21
22 void GPIO_INITIALIZATION (){
23     SET_BIT(APB2ENR, 2);
24     GPIOA_CRL &= 0xFF0FFFFF;
25     GPIOA_CRL |= 0x00000000;
26     GPIOA_CRH &= 0xFF0FFFFF;
27     GPIOA_CRH |= 0x22222222;
28 }
29
```

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

Figure 23 - driver.c

3.1.11 main.c

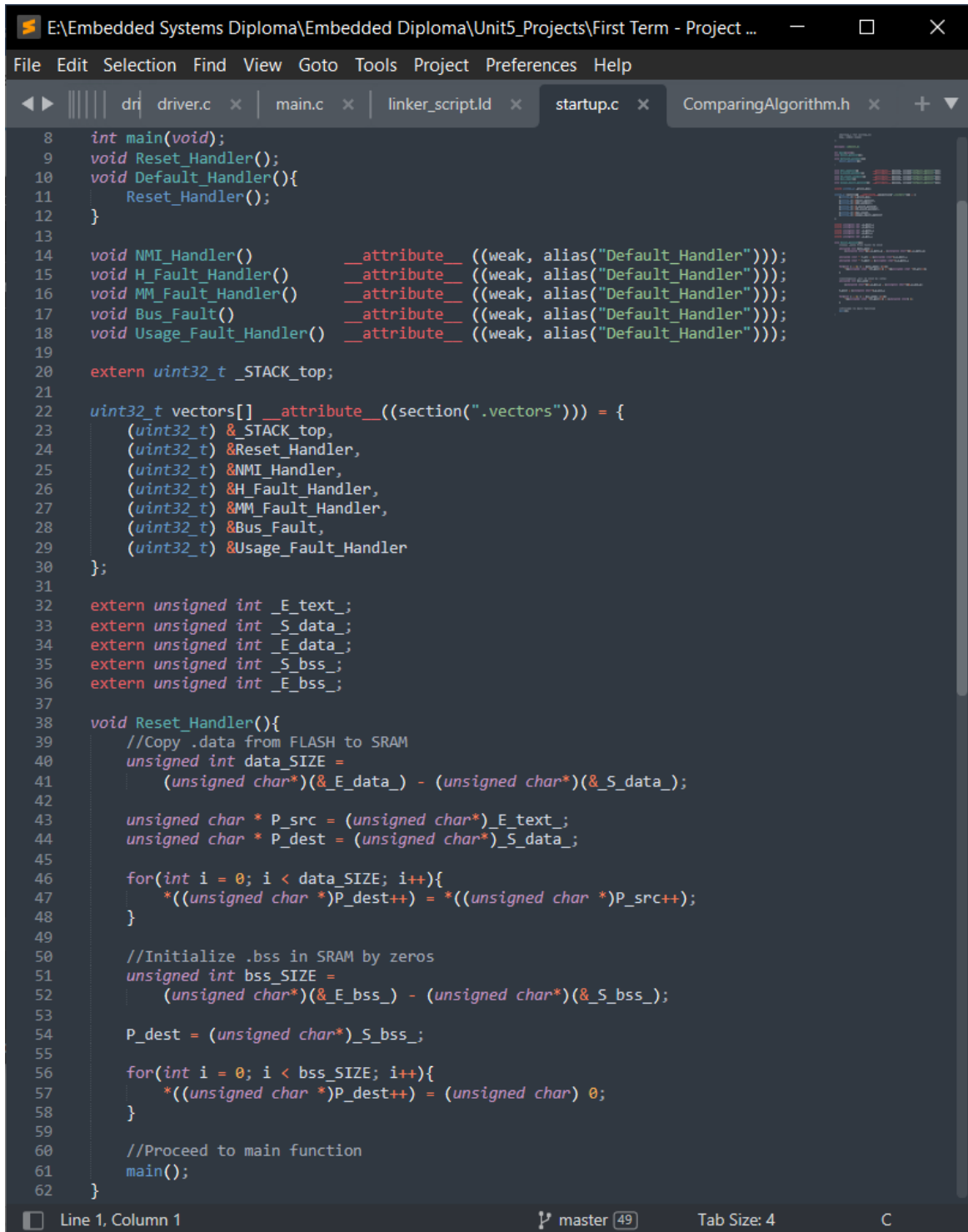


```
1 #include <stdint.h>
2 #include <stdio.h>
3
4 #include "driver.h"
5 #include "PressureSensorDriver.h"
6 #include "ComparingAlgorithm.h"
7 #include "AlarmMonitor.h"
8 #include "AlarmActuatorDriver.h"
9
10 int main (){
11
12     GPIO_INITIALIZATION();
13     PSD_state = STATE(PSD_reading);
14     CA_state = STATE(CA_waiting);
15     AM_state = STATE(AM_alarmOFF);
16     AAD_state = STATE(AAD_waiting);
17
18     while (1)
19     {
20         PSD_state();
21         CA_state();
22         AM_state();
23         AAD_state();
24     }
25
26 }
27
```

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

Figure 24 - main.c

3.1.12 startup.c

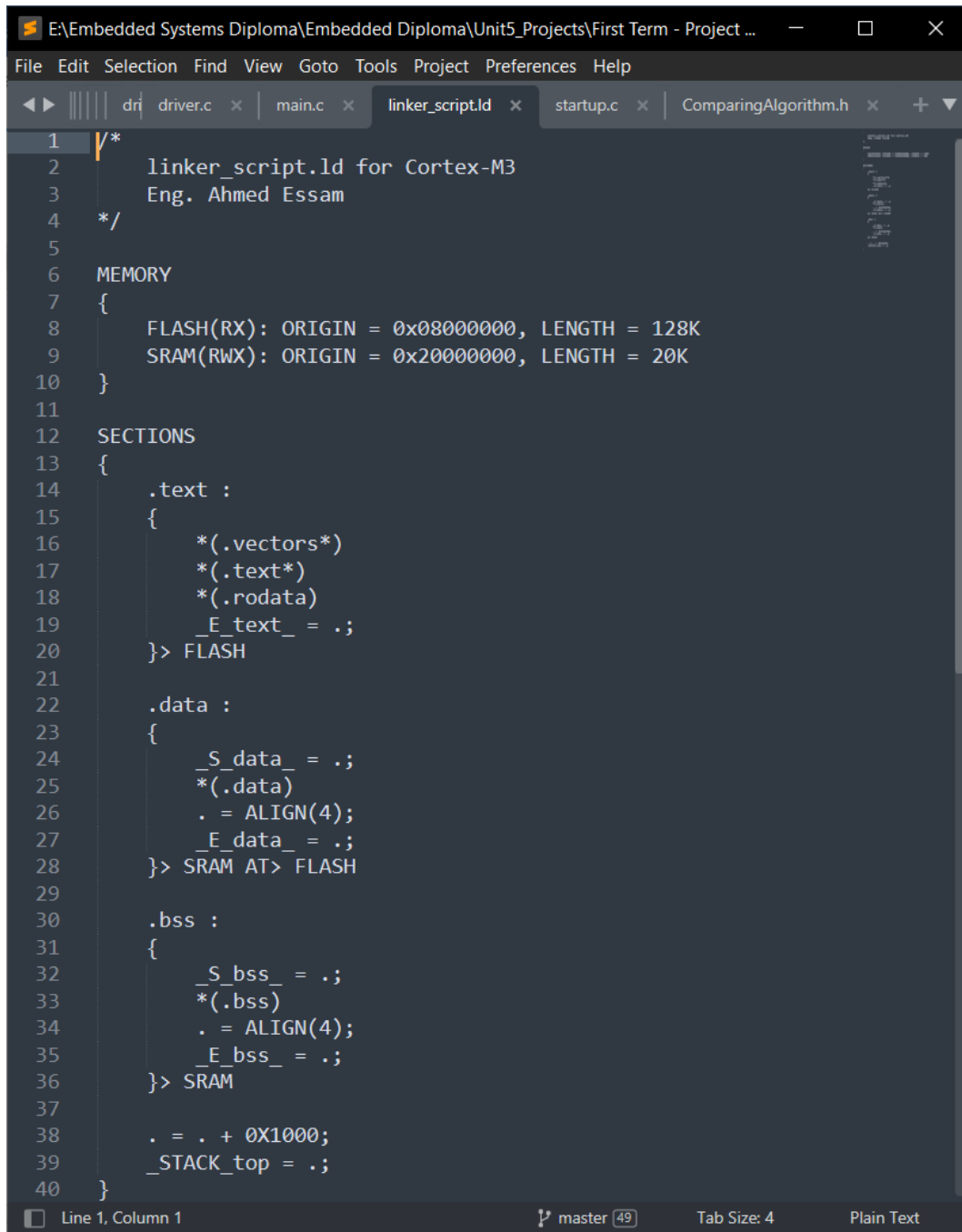


```
8  int main(void);
9  void Reset_Handler();
10 void Default_Handler(){
11     Reset_Handler();
12 }
13
14 void NMI_Handler()      __attribute__((weak, alias("Default_Handler")));
15 void H_Fault_Handler()  __attribute__((weak, alias("Default_Handler")));
16 void MM_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
17 void Bus_Fault()        __attribute__((weak, alias("Default_Handler")));
18 void Usage_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
19
20 extern uint32_t _STACK_top;
21
22 uint32_t vectors[] __attribute__((section(".vectors"))) = {
23     (uint32_t) &_STACK_top,
24     (uint32_t) &Reset_Handler,
25     (uint32_t) &NMI_Handler,
26     (uint32_t) &H_Fault_Handler,
27     (uint32_t) &MM_Fault_Handler,
28     (uint32_t) &Bus_Fault,
29     (uint32_t) &Usage_Fault_Handler
30 };
31
32 extern unsigned int _E_text;
33 extern unsigned int _S_data;
34 extern unsigned int _E_data;
35 extern unsigned int _S_bss;
36 extern unsigned int _E_bss;
37
38 void Reset_Handler(){
39     //Copy .data from FLASH to SRAM
40     unsigned int data_SIZE =
41         (unsigned char*)&_E_data - (unsigned char*)&_S_data;
42
43     unsigned char * P_src = (unsigned char*)&_E_text;
44     unsigned char * P_dest = (unsigned char*)&_S_data;
45
46     for(int i = 0; i < data_SIZE; i++){
47         *((unsigned char *)P_dest++) = *((unsigned char *)P_src++);
48     }
49
50     //Initialize .bss in SRAM by zeros
51     unsigned int bss_SIZE =
52         (unsigned char*)&_E_bss - (unsigned char*)&_S_bss;
53
54     P_dest = (unsigned char*)&_S_bss;
55
56     for(int i = 0; i < bss_SIZE; i++){
57         *((unsigned char *)P_dest++) = (unsigned char) 0;
58     }
59
60     //Proceed to main function
61     main();
62 }
```

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

Figure 25 - startup.c

3.1.13 linker_script.ld

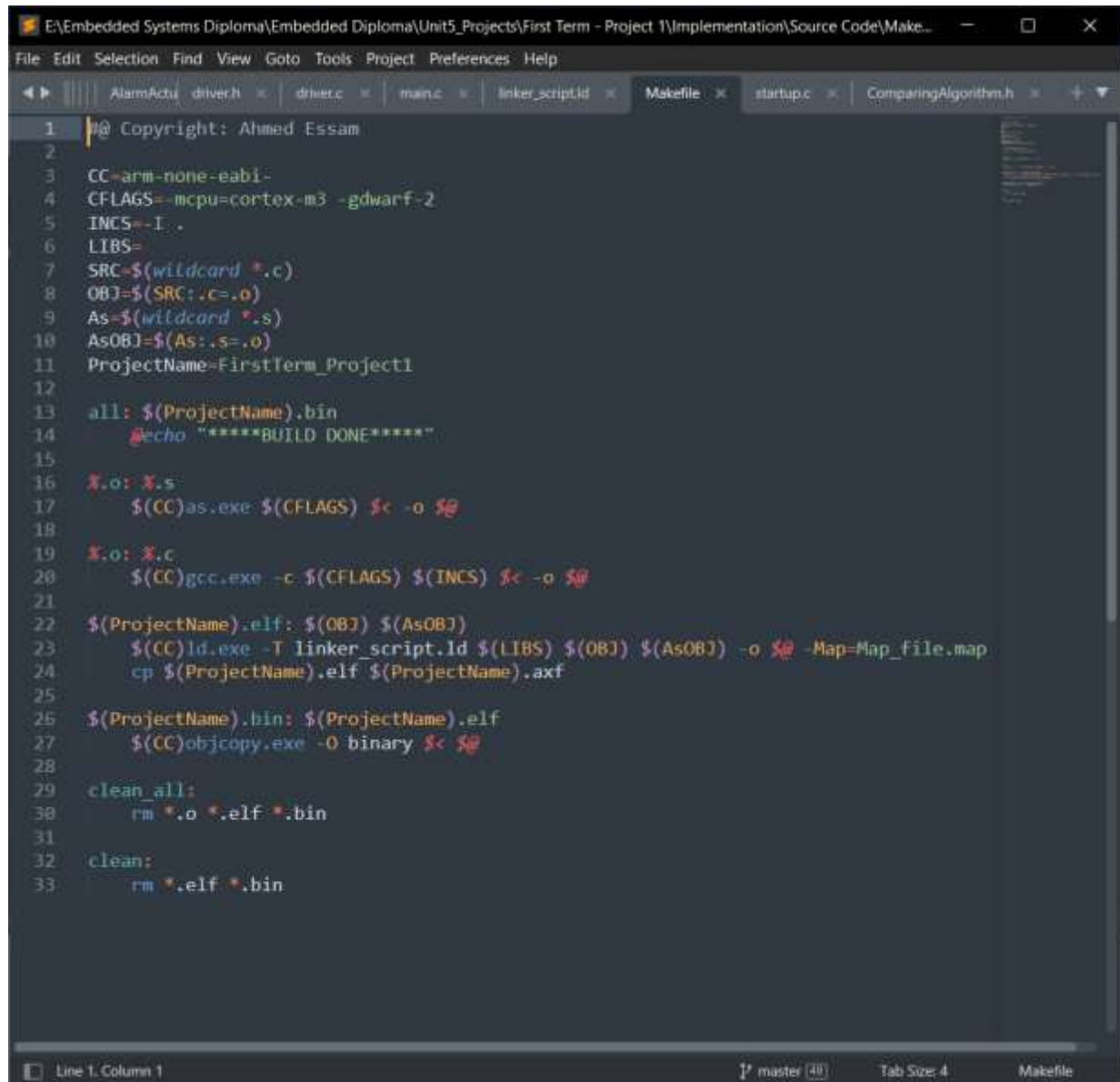


```
1  /*
2     linker_script.ld for Cortex-M3
3     Eng. Ahmed Essam
4  */
5
6  MEMORY
7  {
8      FLASH(RX): ORIGIN = 0x08000000, LENGTH = 128K
9      SRAM(RWX): ORIGIN = 0x20000000, LENGTH = 20K
10 }
11
12 SECTIONS
13 {
14     .text :
15     {
16         *(.vectors*)
17         *(.text*)
18         *(.rodata)
19         _E_text_ = .;
20     }> FLASH
21
22     .data :
23     {
24         _S_data_ = .;
25         *(.data)
26         . = ALIGN(4);
27         _E_data_ = .;
28     }> SRAM AT> FLASH
29
30     .bss :
31     {
32         _S_bss_ = .;
33         *(.bss)
34         . = ALIGN(4);
35         _E_bss_ = .;
36     }> SRAM
37
38     . = . + 0X1000;
39     _STACK_top = .;
40 }
```

Line 1, Column 1 master 49 Tab Size: 4 Plain Text

Figure 26 - linker_script.ld

3.1.14 Makefile



The image shows a screenshot of a code editor window with a dark theme. The title bar at the top reads "E:\Embedded Systems Diploma\Embedded Diploma\Unit5_Projects\First Term - Project 1\Implementation\Source Code\Make...". The menu bar includes "File", "Edit", "Selection", "Find", "View", "Goto", "Tools", "Project", "Preferences", and "Help". The tab bar shows several open files: "AlarmActu...", "driver.h", "driver.c", "main.c", "linker_script.ld", "Makefile" (which is the active tab), "startup.c", and "ComparingAlgorithm.h". The main editor area displays the content of the "Makefile" file, which is a text file with various build rules and variables. The status bar at the bottom indicates "Line 1, Column 1", "master (40)", "Tab Size: 4", and "Makefile".

```
1  ## Copyright: Ahmed Essam
2
3  CC=arm-none-eabi-
4  CFLAGS=-mcpu=cortex-m3 -gdwarf-2
5  INCS=-I.
6  LIBS=
7  SRC=$(wildcard *.c)
8  OBJ=$(SRC:.c=.o)
9  AS=$(wildcard *.s)
10 ASOBJ=$(AS:.s=.o)
11 ProjectName=FirstTerm_Project1
12
13 all: $(ProjectName).bin
14     @echo "*****BUILD DONE*****"
15
16 %.o: %.s
17     $(CC)as.exe $(CFLAGS) $< -o $@
18
19 %.o: %.c
20     $(CC)gcc.exe -c $(CFLAGS) $(INCS) $< -o $@
21
22 $(ProjectName).elf: $(OBJ) $(ASOBJ)
23     $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(ASOBJ) -o $@ -Map=Map_file.map
24     cp $(ProjectName).elf $(ProjectName).axf
25
26 $(ProjectName).bin: $(ProjectName).elf
27     $(CC)objcopy.exe -O binary $< $@
28
29 clean_all:
30     rm *.o *.elf *.bin
31
32 clean:
33     rm *.elf *.bin
```

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.

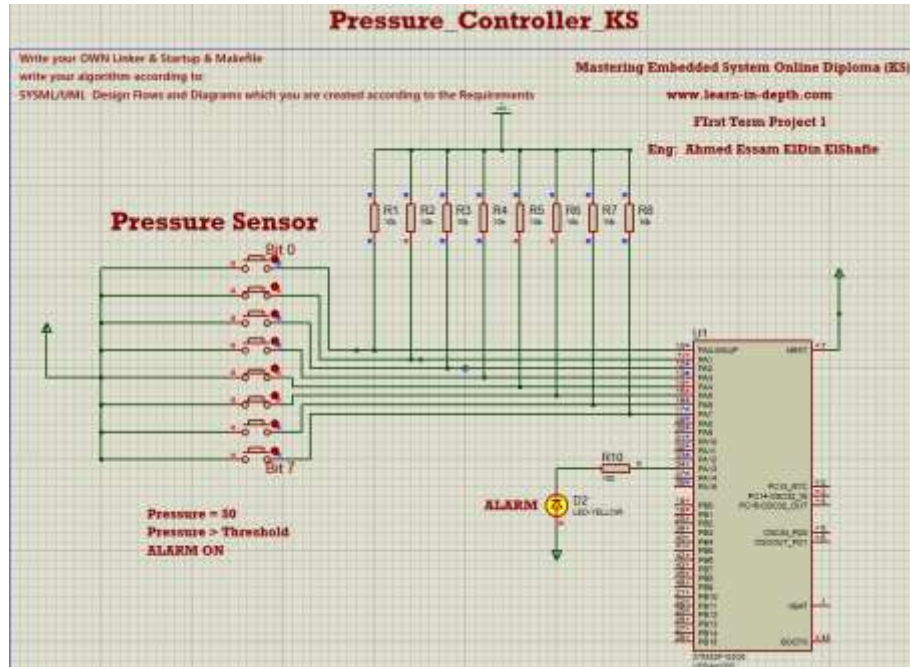


Figure 28 - Pressure = 50 Simulation

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

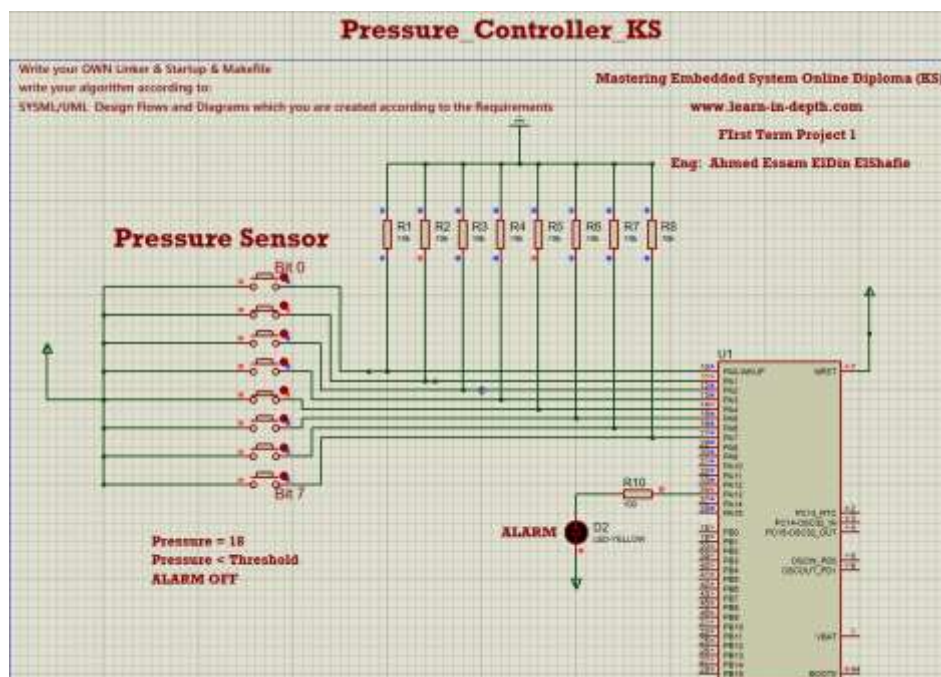


Figure 29 - Pressure = 18 Simulation

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

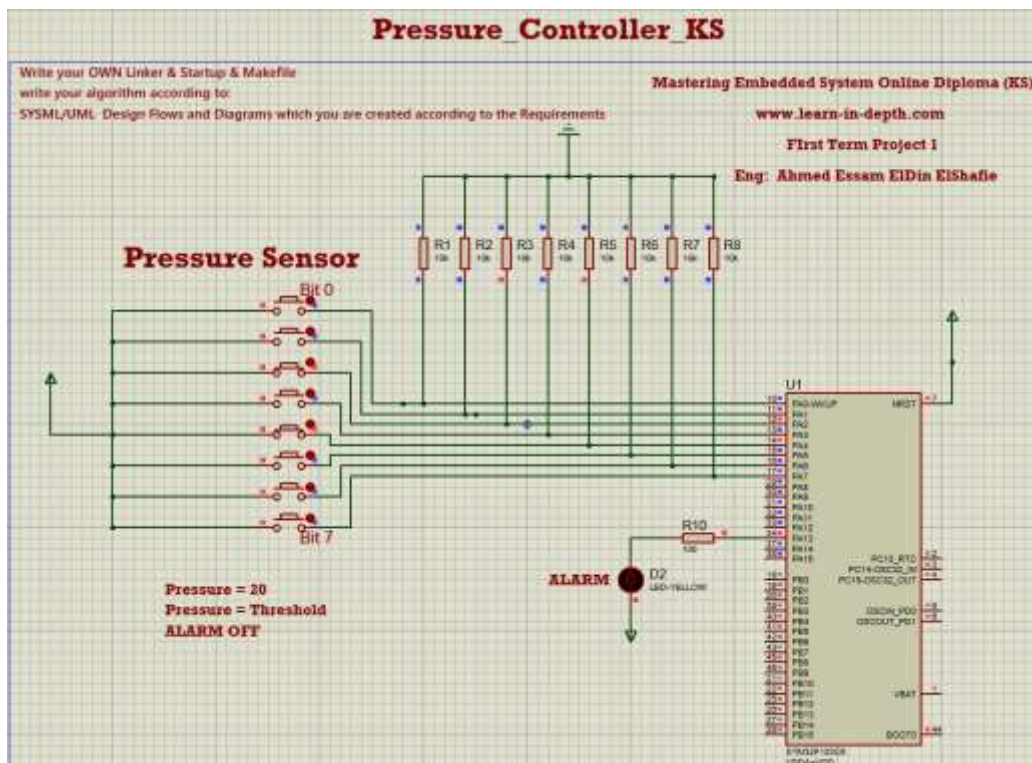


Figure 30 - Pressure = 18 Simulation

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

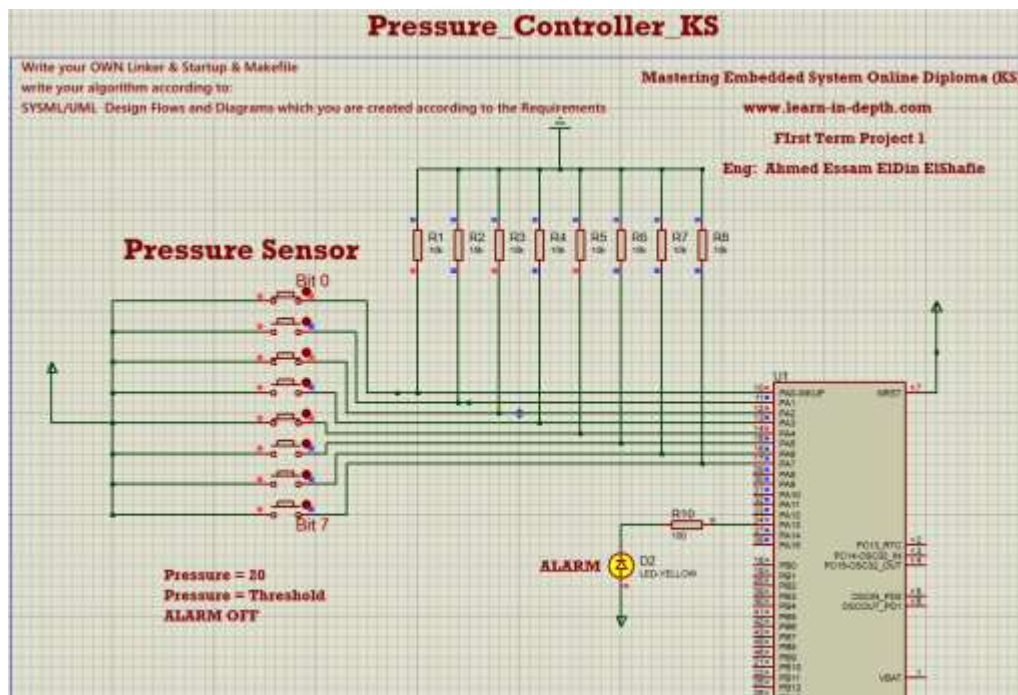


Figure 31 - Pressure = 21 Simulation

3.2.2 Output Mapfile

```
14 Memory Configuration
15
16 Name      Origin      Length      Attributes
17 FLASH     0x08000000  0x00020000  xr
18 SRAM      0x20000000  0x00050000  xrw
19 *default* 0x00000000  0xffffffff
20
21 Linker script and memory map
22
23
24 .text      0x08000000  0x3fc
25 *(.vectors*)
26 .vectors   0x08000000  0x1c startup.o
27           0x08000000  vectors
28
29 *(.text*)
30 .text      0x0800001c  0x98 startup.o
31           0x0800001c  H_Fault_Handler
32           0x0800001c  MM_Fault_Handler
33           0x0800001c  Usage_Fault_Handler
34           0x0800001c  Bus_Fault
35           0x0800001c  Default_Handler
36           0x0800001c  NMI_Handler
37           0x08000028  Reset_Handler
38
39 .text      0x080000b4  0x5c main.o
40           0x080000b4  main
41
42 .text      0x08000110  0xac AlarmActuatorDriver.o
43           0x08000110  startAlarm
44           0x0800012c  stopAlarm
45           0x08000148  aDriver_init
46           0x08000154  State_AAD_alarmOFF
47           0x0800017c  State_AAD_alarmON
48           0x080001a4  State_AAD_waiting
49
50 .text      0x080001bc  0x78 ComparingAlgorithm.o
51           0x080001bc  setPressureVal
52           0x080001e8  State_CA_detect
53           0x0800021c  State_CA_waiting
54
55 .text      0x08000234  0xc4 driver.o
56           0x08000234  Delay
57           0x08000254  getPressureVal
58           0x0800026c  Set_Alarm_actuator
59           0x080002a8  GPIO_INITIALIZATION
60
61 .text      0x080002f8  0x88 AlarmMonitor.o
62           0x080002f8  highPressureFlag
63           0x08000314  State_AM_alarmOFF
64           0x0800032c  State_AM_alarmON
65           0x08000350  State_AM_waiting
66
67 .text      0x08000380  0x70 PressureSensorDriver.o
68           0x08000380  pDriver_init
69           0x0800038c  State_PSD_reading
70           0x080003c4  State_PSD_waiting
71
72 *(.rodata)
73 .rodata    0x080003f0  0x4 ComparingAlgorithm.o
74           0x080003f0  threshold
75
76 .rodata    0x080003f4  0x4 AlarmMonitor.o
77           0x080003f4  aTimer
```

Figure 32 - Mapfile 1/2

```

68      aTimer
69      .rodata      0x080003f8      0x4 PressureSensorDriver.o
70      0x080003f8      pTimer
71      0x080003fc      _E_text_ = .
72
73      .glue_7      0x080003fc      0x0
74      .glue_7      0x080003fc      0x0 linker stubs
75
76      .glue_7t     0x080003fc      0x0
77      .glue_7t     0x080003fc      0x0 linker stubs
78
79      .vfp11_veneer 0x080003fc      0x0
80      .vfp11_veneer 0x080003fc      0x0 linker stubs
81
82      .v4_bx       0x080003fc      0x0
83      .v4_bx       0x080003fc      0x0 linker stubs
84
85      .iplt        0x080003fc      0x0
86      .iplt        0x080003fc      0x0 startup.o
87
88      .rel.dyn     0x080003fc      0x0
89      .rel.iplt    0x080003fc      0x0 startup.o
90
91      .data        0x20000000      0x0 load address 0x080003fc
92      0x20000000      _S_data_ = .
93      *(.data)
94      .data        0x20000000      0x0 startup.o
95      .data        0x20000000      0x0 main.o
96      .data        0x20000000      0x0 AlarmActuatorDriver.o
97      .data        0x20000000      0x0 ComparingAlgorithm.o
98      .data        0x20000000      0x0 driver.o
99      .data        0x20000000      0x0 AlarmMonitor.o
100     .data        0x20000000      0x0 PressureSensorDriver.o
101     0x20000000      . = ALIGN (0x4)
102     0x20000000      _E_data_ = .
103
104     .igot.plt     0x20000000      0x0 load address 0x080003fc
105     .igot.plt     0x20000000      0x0 startup.o
106
107     .bss         0x20000000      0x25 load address 0x080003fc
108     0x20000000      _S_bss_ = .
109     *(.bss)
110     .bss         0x20000000      0x0 startup.o
111     .bss         0x20000000      0x0 main.o
112     .bss         0x20000000      0x0 AlarmActuatorDriver.o
113     .bss         0x20000000      0x4 ComparingAlgorithm.o
114     0x20000000      CA_pVal
115     .bss         0x20000004      0x0 driver.o
116     .bss         0x20000004      0x0 AlarmMonitor.o
117     .bss         0x20000004      0x4 PressureSensorDriver.o
118     0x20000004      PSD_pVal
119     0x20000008      . = ALIGN (0x4)
120     0x20000008      _E_bss_ = .
121     COMMON       0x20000008      0x5 AlarmActuatorDriver.o
122     0x20000008      AAD_state

```

Figure 33 - Mapfile 2/2