

Mastering Embedded Systems Online Diploma

www.learn-in-depth.com

First-term (Final Project 1)

Eng. Mostafa Mahmoud Helmy Abd-elrahman

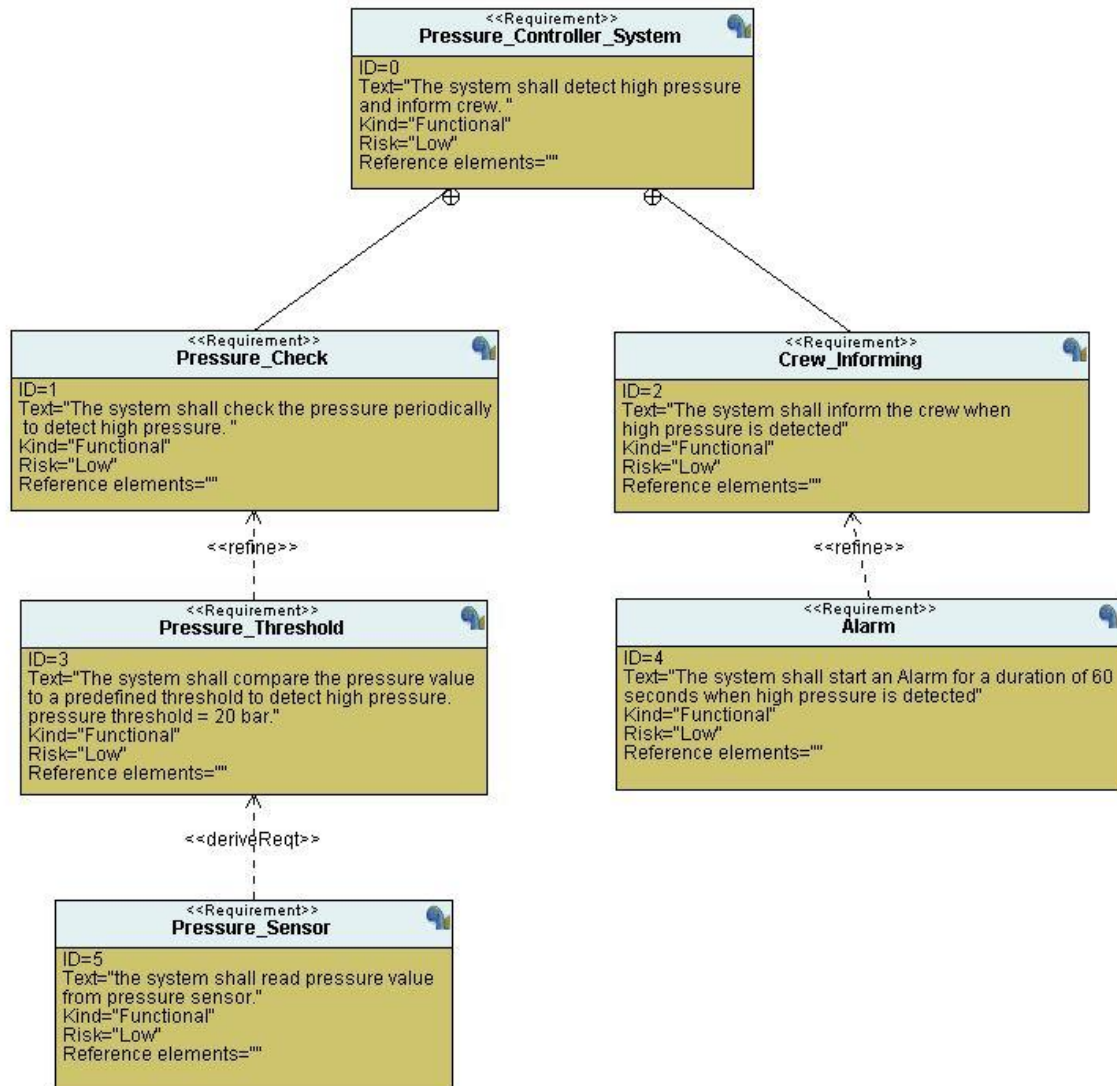
My Profile:

www.learn-in-depth.com/online-diploma/darshme7@gmail.com

Pressure Controller Project

- **Case study: Pressure controller system**
 - Pressure controller system should check if the pressure inside a cabin exceeds 20 bar.
 - If the pressure inside a cabin exceeds 20 bar the pressure controller system should inform the crew with an alarm for a duration of 60 seconds.
- **Assumptions:**
 - System setup and shutdown procedure are not modelled.
 - System maintenance is not modelled.
 - Pressure sensor never fails.
 - Alarm actuator never fails.
 - System never faces power cut.
- **Method: for Pressure Controller software development cycle** [Waterfall Model](#) was found to be the most suitable.

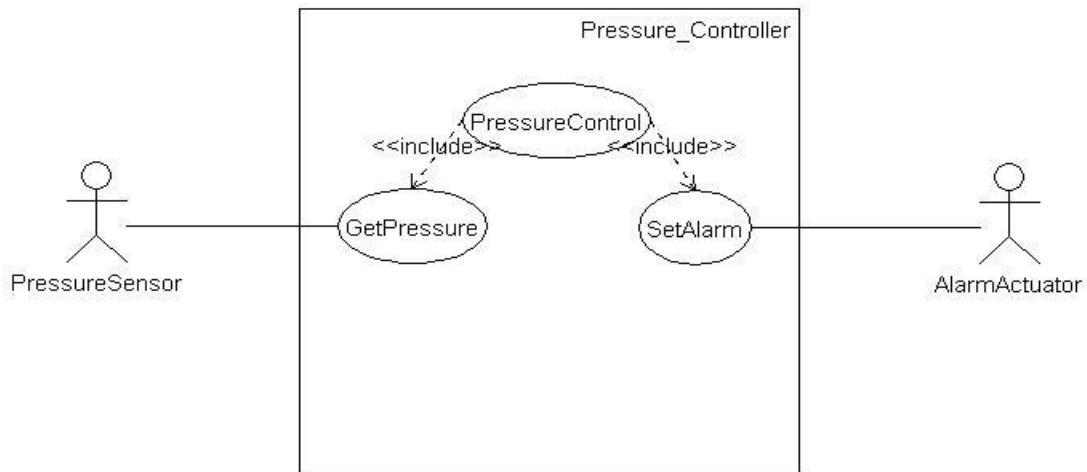
- Requirement diagram:



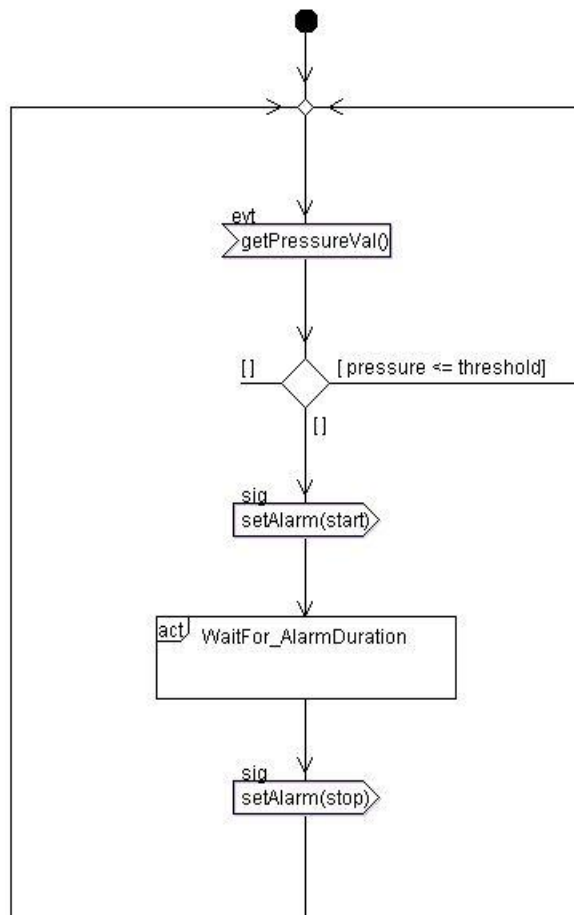
- Space exploration/Partitioning: the project is quite simple, it doesn't require more than one ECU and STM32 was found suitable for this project.

- **System analysis:**

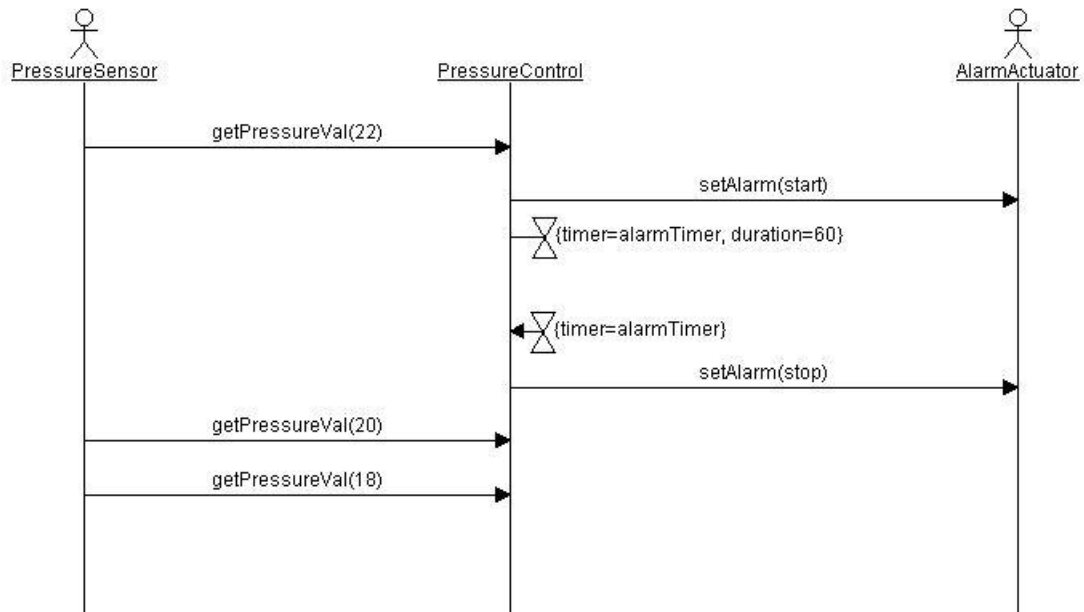
- **Use Case diagram:**



- **Activity diagram:**

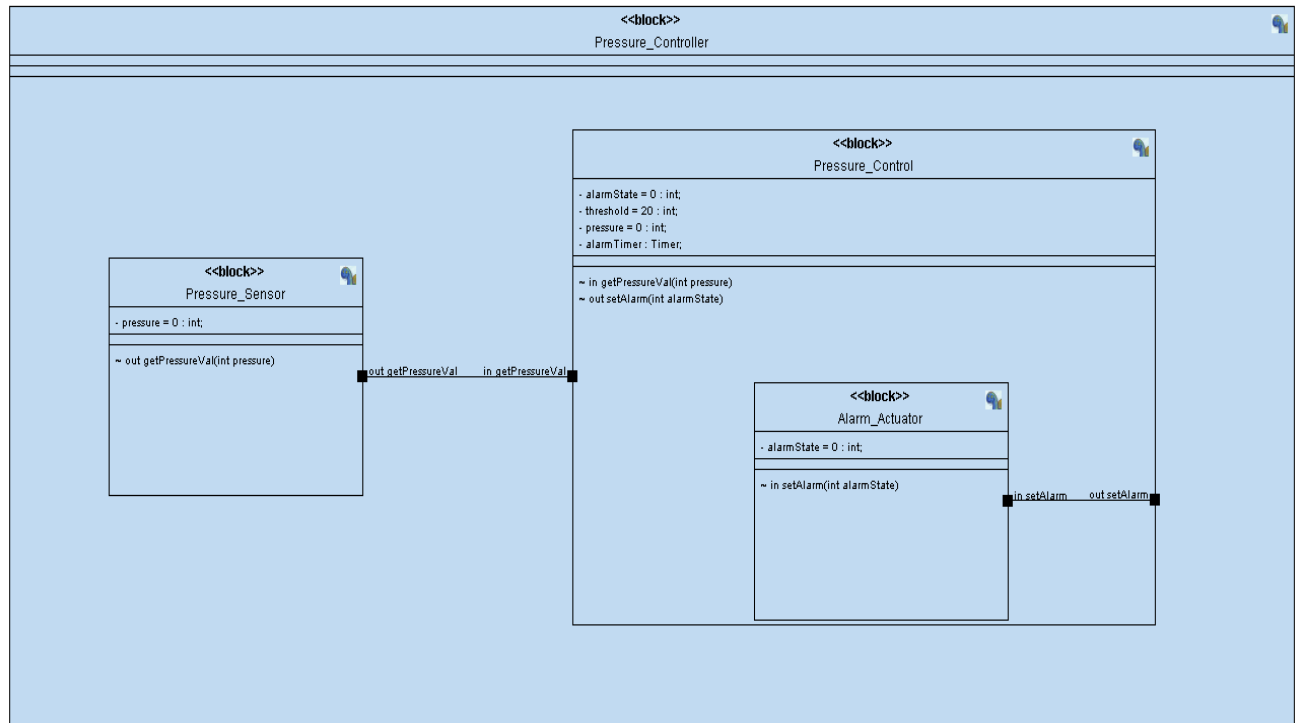


- **Sequence diagram:**

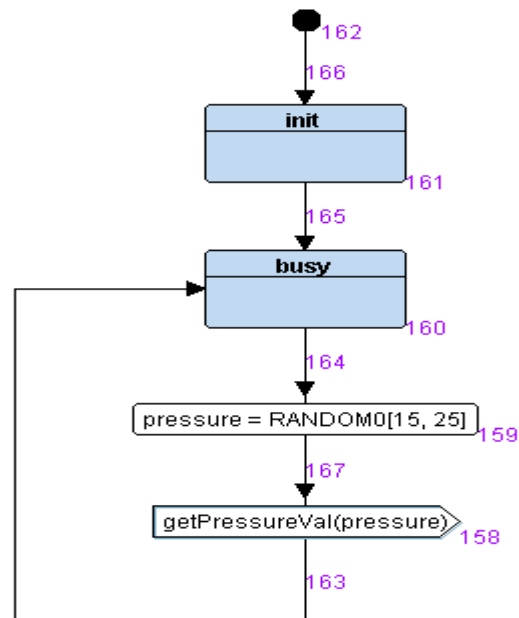


- **System Design:**

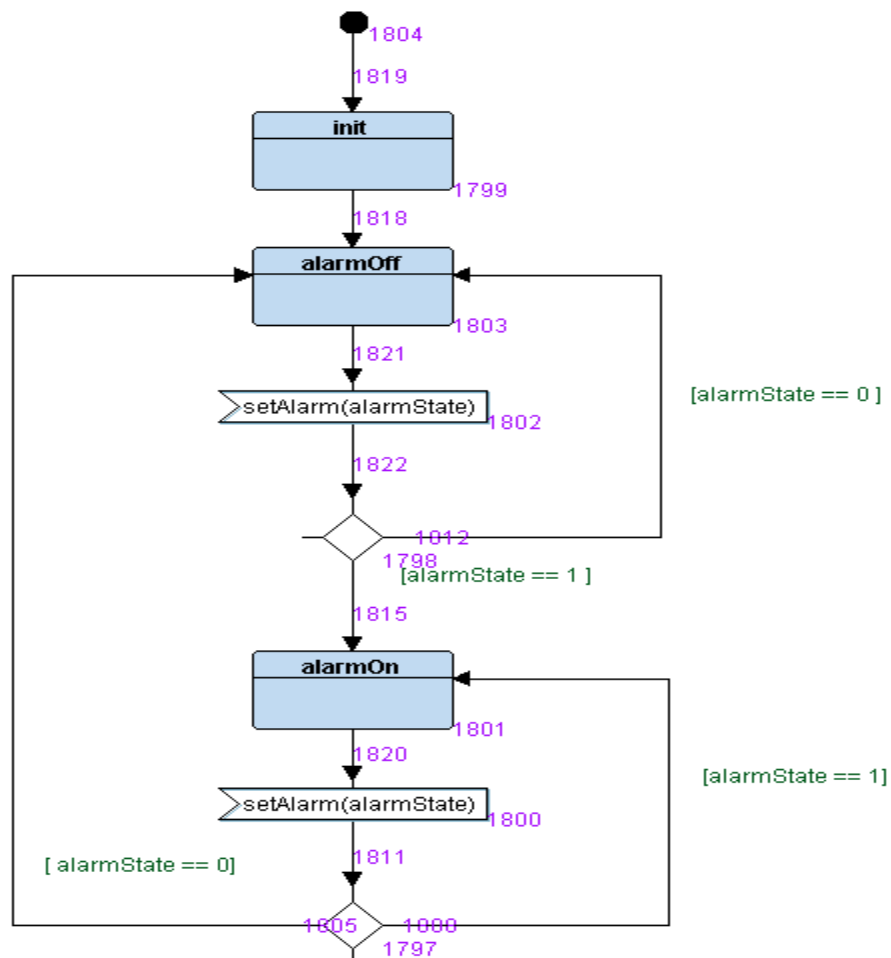
- **Block diagram:**



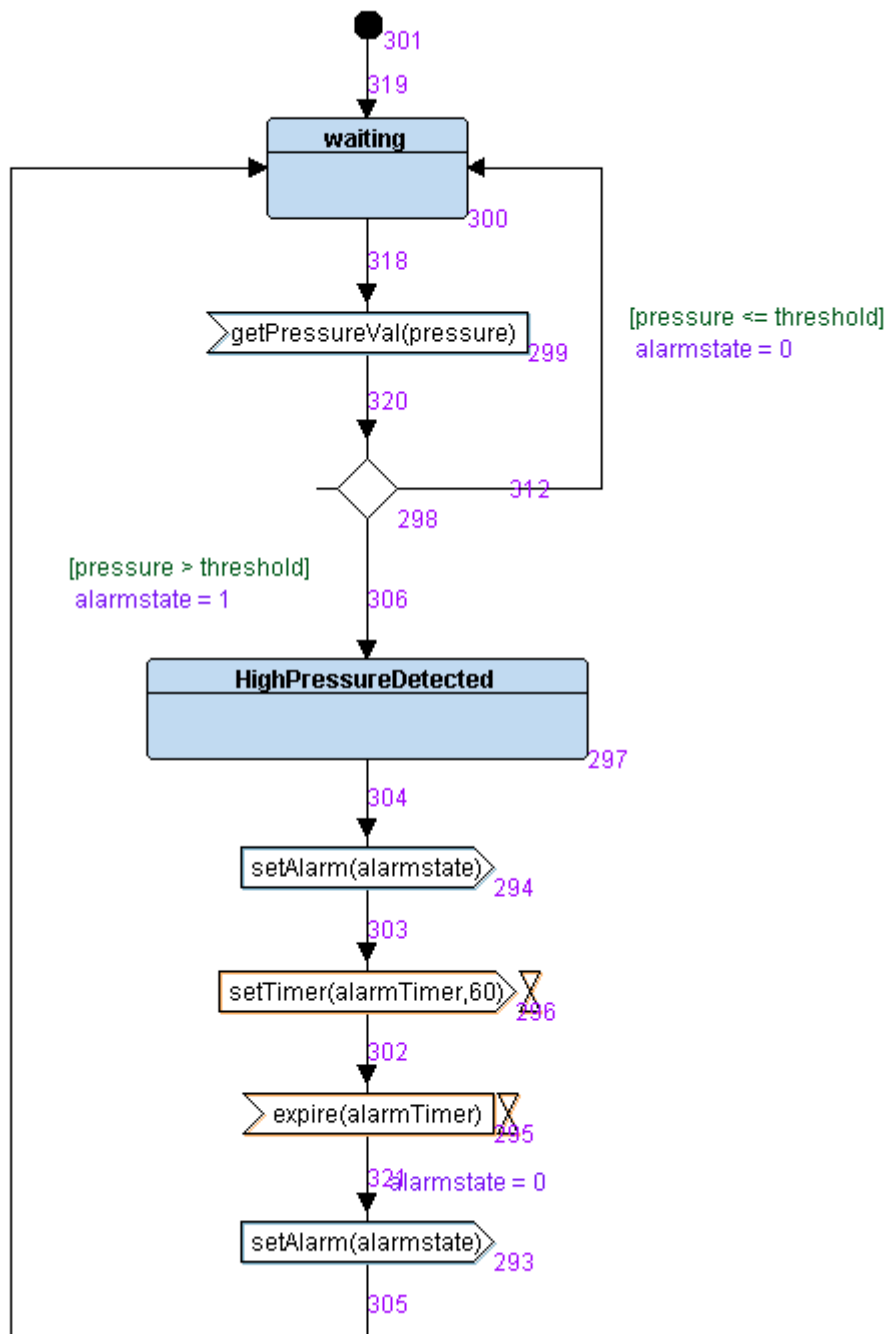
○ State Machine Pressure_Sensor:



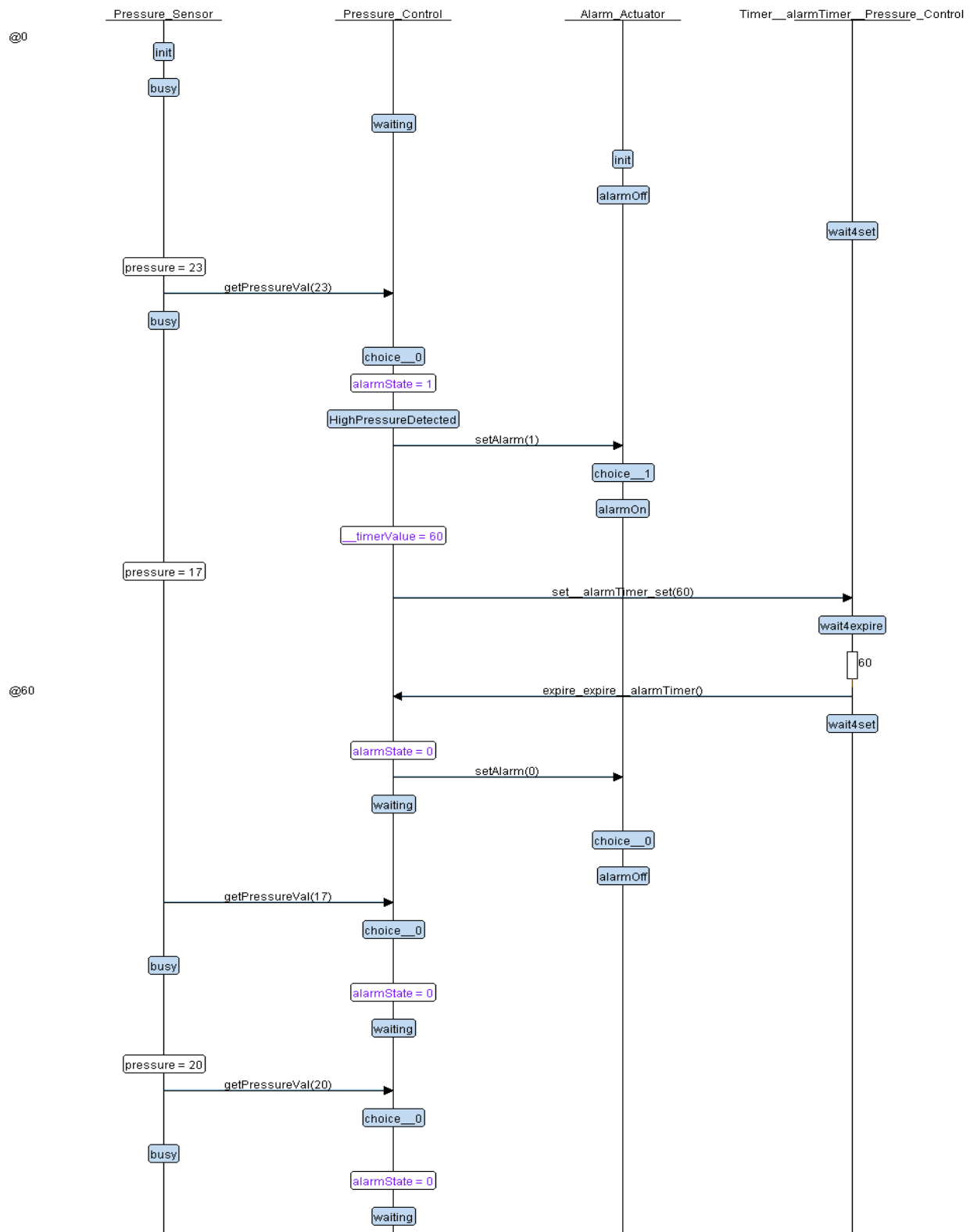
○ State Machine Alarm_Actuator:



○ State Machine Pressure_Control:



- Simulation:



- **Software Implementation of project for each module:**

- **Pressure_Controller main.c:**

```

1  /*
2   * main.c
3   *
4   * Created on: May 9, 2021
5   * Author: mostafa
6   */
7
8  #include "PS.h"
9  #include "PC.h"
10
11
12 void setup(void) {
13
14     GPIO_INITIALIZATION();
15     PS_state=STATE(PS_busy);
16     PC_state=STATE(PC_waiting);
17
18 }
19
20 int main(void) {
21
22     setup();
23     while(1) {
24
25         PS_state();
26         PC_state();
27     }
28
29     return 0;
30 }

```

- **Pressure_Controller states.h:**

```

1  /*
2   * states.h
3   *
4   * Created on: May 9, 2021
5   * Author: mostafa
6   */
7
8  #ifndef STATES_H_
9  #define STATES_H_
10
11 #include "Platform_Types.h"
12 #include "driver.h"
13
14
15 #define STATE_define(_statFUN_) void ST_##_statFUN_()
16 #define STATE(_statFUN_) ST_##_statFUN_
17
18 void getPressureValue(uint32 p);
19 void setAlarm(uint32 st);
20
21 #endif /* STATES_H_ */
22

```

- **Pressure_Sensor PS.c:**

```
1  /*
2      * PS.c
3      *
4      * Created on: May 9, 2021
5      * Author: mostafa
6      */
7
8      #include "PS.h"
9
10
11     void (*PS_state) ();
12
13     uint32 PS_pressure = 0;
14
15     STATE_define(PS_busy) {
16
17         PS_state_id = PS_busy;
18
19         PS_pressure = getPressureVal();
20
21         getPressureValue(PS_pressure);
22
23         PS_state=STATE(PS_busy);
24     }
25
```

- **Pressure_Sensor PS.h:**

```
1  /*
2      * PS.h
3      *
4      * Created on: May 9, 2021
5      * Author: mostafa
6      */
7
8     #ifndef PS_H_
9     #define PS_H_
10
11     #include "states.h"
12
13     enum{
14         PS_busy
15     }PS_state_id;
16
17     STATE_define(PS_busy);
18
19
20     extern void (*PS_state) ();
21
22     #endif /* PS_H_ */
23
```

○ Pressure_Control PC.c:

```
8  #include "PC.h"
9  #include "AA.h"
10
11  void (*PC_state)();
12
13  uint32 PC_pressure = 0;
14  uint32 PC_threshold = 20;
15  enum {
16      ON,
17      OFF
18  } PC_alarmstate;
19
20  void getPressureValue(uint32 p){
21
22      PC_pressure = p;
23
24      (PC_pressure > PC_threshold)?(PC_state=STATE(PC_HPdetected)):(PC_state=STATE(PC_waiting));
25  }
26
27  STATE_define(PC_waiting){
28
29      PC_state_id = PC_waiting;
30      PC_alarmstate = OFF;
31      setAlarm(PC_alarmstate);
32      AA_state();
33  }
34
35  STATE_define(PC_HPdetected){
36
37      PC_state_id = PC_HPdetected;
38      PC_alarmstate = ON;
39      setAlarm(PC_alarmstate);
40      AA_state();
41
42      Delay(6000000);
43
44      PC_alarmstate = OFF;
45      setAlarm(PC_alarmstate);
46      AA_state();
47
48      PC_state = STATE(PC_waiting);
49  }
50
```

○ Presure_Control PC.h:

```
1  /*
2   * PC.h
3   *
4   * Created on: May 9, 2021
5   * Author: mostafa
6   */
7
8  #ifndef PC_H_
9  #define PC_H_
10
11  #include "states.h"
12
13  enum {
14      PC_waiting,
15      PC_HPdetected
16  } PC_state_id;
17
18  STATE_define(PC_waiting);
19  STATE_define(PC_HPdetected);
20
21  extern void (*PC_state)();
22
23  #endif /* PC_H_ */
24
```

○ Alarm_Actuator AA.c:

```
1  /*
2   * AA.c
3   *
4   * Created on: May 9, 2021
5   * Author: mostafa
6   */
7  #include "AA.h"
8
9  void (*AA_state) ();
10
11  enum {
12      ON,
13      OFF
14  } AA_alarmstate;
15
16  void setAlarm(uint32 st){
17
18      AA_alarmstate = st;
19
20      (AA_alarmstate == ON)?(AA_state = STATE(AA_alarmOn)):(AA_state = STATE(AA_alarmOff));
21  }
22
23
24
25  STATE_define(AA_alarmOn){
26
27      AA_state_id = AA_alarmON;
28
29      Set_Alarm_actuator(AA_alarmstate);
30  }
31
32
33  STATE_define(AA_alarmOff){
34
35      AA_state_id = AA_alarmOFF;
36
37      Set_Alarm_actuator(AA_alarmstate);
38  }
39
40
```

○ Alarm_Actuator AA.h:

```
1  /*
2   * AA.h
3   *
4   * Created on: May 9, 2021
5   * Author: mostafa
6   */
7
8
9  #ifndef AA_H_
10 #define AA_H_
11
12 #include "states.h"
13
14 enum{
15     AA_alarmON,
16     AA_alarmOFF
17 }AA_state_id;
18
19 STATE_define(AA_alarmOn);
20 STATE_define(AA_alarmOff);
21
22 extern void (*AA_state) ();
23 #endif /* AA_H_ */
24
```


- **Simulation on proteus:**

- **Case 1:**

- **Pressure_Sensor (PS) Module sent the reading(18) to Pressure_Control (PC) Module to compare it with threshold(20), in this case alarm is kept off as $(18 < 20)$**

Proteus 8 Professional - Schematic Capture

File Edit View Tool Design Graph Debug Library Template System Help

Base Design

Schematic Capture

CM3 Variables - U1

Name	Address	Value
AA_alarmstate	20001011	OFF (1)
PS_state_id	20001012	PS_busy (0)
PC_state_id	20001013	PC_waiting (0)
PC_state_id	20001013	PC_waiting (0)
AA_state_id	20001010	AA_alarmOFF (1)
PC_pressure	20000004	18
PC_threshold	20000000	20
PC_alarmstate	20001014	OFF (1)
PS_state_id	20001012	PS_busy (0)
PS_pressure	20000008	18
vectors	08000000	dword[7]
vectors[0]	08000000	536875020
vectors[1]	08000004	134218641
vectors[2]	08000008	134218629
vectors[3]	0800000C	134218629
vectors[4]	08000010	134218629
vectors[5]	08000014	134218629
vectors[6]	08000018	134218629
AA_state_id	20001010	AA_alarmOFF (1)

Pressure_Sensor (PS) Module sent the reading(18) to Pressure_Control (PC) Module to compare it with threshold(20), in this case alarm is not turned on as $(18 < 20)$

Pressure Sensor

First Term Project 1
Eng: Mostafa Mahmoud Helmy

CM3 Source Code - U1

PCc

```

8000218 void getPressureValue(uint32 p){
8000220     PC_pressure = p;
800022C     (PC_pressure > PC_threshold)?(PC_state=STATE(PC_HPdetected)):(PC_state=STATE(PC_waiting));
800026A }
8000274 STATE_define(PC_waiting){
8000278     PC_state_id = PC_waiting;
8000286     PC_alarmstate = OFF;
8000294     setAlarm(PC_alarmstate);
80002A4     AA_state();
80002B0 }
80002B4 STATE_define(PC_HPdetected){
80002B8     PC_state_id = PC_HPdetected;
80002C6     PC_alarmstate = ON;
80002D4     setAlarm(PC_alarmstate);

```

U1

PA0-WUUP NRST

PA1

PA2

PA3

PA4

PA5

PA6

PA7

PA8

PA9

PA10

PA11

PA12

PA13

PA14

PA15

PC13_RTC

PC14-OSC32_IN

PC15-OSC32_OUT

OSCIN_PDO

OSCCOUT_PD1

VBAT

BOOT0

STM32F103C6

VDDA=VDD

VSSA=VSS

R10

100

ALARM

D2

LED-YELLOW

- **Case 2:**
 - **Pressure_Sensor (PS) Module sent the reading(22) to Pressure_Control (PC) Module to compare it with threshold(20), In this case PC Module sets Alarm_Actuator (AA) Module to turn alarm on for 60 seconds as (22 > 20)**

yarb - Proteus 8 Professional - Schematic Capture

File Edit View Tool Design Graph Debug Library Template System Help

Base Design

Schematic Capture X

CM3 Variables - U1

Name	Address	Value
AA_alarmstate	20001011	OFF (1)
PS_state_id	20001012	PS_busy (0)
PC_state_id	20001013	PC_HPDetected (1)
PC_state_id	20001013	PC_HPDetected (1)
AA_state_id	20001010	AA_alarmOFF (1)
PC_pressure	20000004	22
PC_threshold	20000000	20
PC_alarmstate	20001014	OFF (1)
PS_state_id	20001012	PS_busy (0)
PS_pressure	20000008	22
vectors	08000000	dword[7]
vectors[0]	08000000	536875020
vectors[1]	08000004	134218641
vectors[2]	08000008	134218629
vectors[3]	0800000C	134218629
vectors[4]	08000010	134218629
vectors[5]	08000014	134218629
vectors[6]	08000018	134218629
AA_state_id	20001010	AA_alarmOFF (1)

Pressure_Sensor (PS) Module sent the reading(22) to Pressure_Control (PC) Module to compare it with threshold(20), in this case PC Module sets Alarm_Actuator (AA) Module to turn alarm on as (22 > 20)

Pressure Sensor

First Term Project 1
Eng: Mostafa Mahmoud Helmy

CM3 Source Code - U1

PC.c

```

800026A }
8000274 STATE_define(PC_waiting){
8000278     PC_state_id = PC_waiting;
8000286     PC_alarmstate = OFF;
8000294     setAlarm(PC_alarmstate);
80002A4     AA_state();
80002B0 }
80002B4 STATE_define(PC_HPDetected){
80002B8     PC_state_id = PC_HPDetected;
80002C6     PC_alarmstate = ON;
80002D4     setAlarm(PC_alarmstate);
80002E4     AA_state();
80002F0     Delay(6000000);
80002FC     PC_alarmstate = OFF;

```

U1

PA0-WKUP
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PB8
PB9
PB10
PB11
PB12
PB13
PB14
PB15
NRST
PC13_RTC
PC14-OSC32_IN
PC15-OSC32_OUT
OSCIN_F0D
OSCOUT_F0D
VBAT
BOOT0
STM32F103C8
VDDA=VDD
VSSA=VSS

R10 100

ALARM D2 LED-YELLOW

PAUSED: 00:00:50.038374

- **Software analysis:**

- **Symbols table:**

```
1 Pressure_Controller.elf: Symbols
2
3 2000000c B _E_bss
4 20000004 D _E_data
5 08000440 T _E_text
6 20000004 B _S_bss
7 20000000 D _S_data
8 2000100c B _stack_top
9 20001011 B AA_alarmstate
10 2000100c B AA_state
11 20001010 B AA_state_id
12 08000384 W Bus_Fault_Handler
13 08000384 T Default_Handler
14 080000b8 T Delay
15 080000dc T getPressureVal
16 08000218 T getPressureValue
17 08000144 T GPIO_INITIALIZATION
18 08000384 W Hard_Fault_Handler
19 080001f4 T main
20 08000384 W MM_Handler
21 08000384 W NMI_Handler
22 20001014 B PC_alarmstate
23 20000004 B PC_pressure
24 20001018 B PC_state
25 20001013 B PC_state_id
26 20000000 D PC_threshold
27 20000008 B PS_pressure
28 2000101c B PS_state
29 20001012 B PS_state_id
30 08000390 T Reset_Handler
31 080000f4 T Set_Alarm_actuator
32 0800001c T setAlarm
33 080001c4 T setup
34 08000094 T ST_AA_alarmOff
35 08000070 T ST_AA_alarmOn
36 080002b4 T ST_PC_HPdetected
37 08000274 T ST_PC_waiting
38 0800033c T ST_PS_busy
39 08000384 W Usage_Fault_Handler
40 08000000 T vectors
41
```

- **Sections table:**

```

1 Pressure_Controller.elf:      file format elf32-littlearm
2
3 Sections:
4 Idx Name                      Size      VMA      LMA      File off  Algn
5  0 .text                     00000440 08000000 08000000 00008000 2**2
6                                CONTENTS, ALLOC, LOAD, READONLY, CODE
7  1 .data                     00000004 20000000 08000440 00010000 2**2
8                                CONTENTS, ALLOC, LOAD, DATA
9  2 .bss                      0000101c 20000004 08000444 00010004 2**2
10                               ALLOC
11  3 .comment                  00000011 00000000 00000000 00010004 2**0
12                               CONTENTS, READONLY
13  4 .ARM.attributes           00000033 00000000 00000000 00010015 2**0
14                               CONTENTS, READONLY
15
16
17

```

- **Map file:**

```

1 Allocating common symbols
2 Common symbol      size      file
3
4 PC_alarmstate      0x1      PC.o
5 AA_state            0x4      AA.o
6 PC_state            0x4      PC.o
7 AA_state_id         0x1      AA.o
8 PS_state            0x4      PS.o
9 PS_state_id         0x1      main.o
10 PC_state_id         0x1      main.o
11 AA_alarmstate       0x1      AA.o
12
13 Memory Configuration
14
15 Name              Origin              Length              Attributes
16 flash              0x08000000          0x00020000          xr
17 sram                0x20000000          0x00005000          xrw
18 *default*          0x00000000          0xffffffff
19

```


20 Linker script and memory map

21

22

23 .text 0x08000000 0x440

24 *(.vectors)

25 .vectors 0x08000000 0x1c startup.o

26 0x08000000 vectors

27 *(.text*)

28 .text 0x0800001c 0x9c AA.o

29 0x0800001c setAlarm

30 0x08000070 ST_AA_alarmOn

31 0x08000094 ST_AA_alarmOff

32 .text 0x080000b8 0x10c driver.o

33 0x080000b8 Delay

34 0x080000dc getPressureVal

35 0x080000f4 Set_Alarm_actuator

36 0x08000144 GPIO_INITIALIZATION

37 .text 0x080001c4 0x54 main.o

38 0x080001c4 setup

39 0x080001f4 main

40 .text 0x08000218 0x124 PC.o

41 0x08000218 getPressureValue

42 0x08000274 ST_PC_waiting

43 0x080002b4 ST_PC_HPdetected

44 .text 0x0800033c 0x48 PS.o

45 0x0800033c ST_PS_busy

46 .text 0x08000384 0xbc startup.o

47 0x08000384 MM_Handler

48 0x08000384 Bus_Fault_Handler

49 0x08000384 Hard_Fault_Handler

50 0x08000384 Default_Handler

51 0x08000384 Usage_Fault_Handler

52 0x08000384 NMI_Handler

53 0x08000390 Reset_Handler

54 *(.rodata)

55 0x08000440 _E_text = .

56

75	.data	0x20000000	0x4 load address 0x08000440
76		0x20000000	_S_data = .
77	*(.data*)		
78	.data	0x20000000	0x0 AA.o
79	.data	0x20000000	0x0 driver.o
80	.data	0x20000000	0x0 main.o
81	.data	0x20000000	0x4 PC.o
82		0x20000000	PC_threshold
83	.data	0x20000004	0x0 PS.o
84	.data	0x20000004	0x0 startup.o
85		0x20000004	. = ALIGN (0x4)
86		0x20000004	_E_data = .
87			
88	.igot.plt	0x20000004	0x0 load address 0x08000444
89	.igot.plt	0x00000000	0x0 AA.o
90			
91	.bss	0x20000004	0x101c load address 0x08000444
92		0x20000004	_S_bss = .
93	*(.bss*)		
94	.bss	0x20000004	0x0 AA.o
95	.bss	0x20000004	0x0 driver.o
96	.bss	0x20000004	0x0 main.o
97	.bss	0x20000004	0x4 PC.o
98		0x20000004	PC_pressure
99	.bss	0x20000008	0x4 PS.o
100		0x20000008	PS_pressure
101	.bss	0x2000000c	0x0 startup.o
102		0x2000000c	. = ALIGN (0x4)
103		0x2000000c	_E_bss = .
104		0x2000100c	. = (. + 0x1000)
105	*fill*	0x2000000c	0x1000
106		0x2000100c	_stack_top = .
107	COMMON	0x2000100c	0x6 AA.o
108		0x2000100c	AA_state
109		0x20001010	AA_state_id
110		0x20001011	AA_alarmstate
111	COMMON	0x20001012	0x2 main.o
112		0x20001012	PS_state_id
113		0x20001013	PC_state_id
114	COMMON	0x20001014	0x8 PC.o
115		0x20001014	PC_alarmstate
116		0x20001018	PC_state
117	COMMON	0x2000101c	0x4 PS.o
118		0x2000101c	PS_state