

Pressure Detection

Report



Mastering Embedded System Online Diploma

Prepared By : Sherif Ashraf Khedr

My Profile Link

Table Of Content

1 - Client Requirement	4
1.1 - Requirement Diagram	5
2 - Methodology	6
3 - System Analysis	7
3.1 - Use Case Diagram	7
3.2 - Activity Diagram	8
3.3 - Sequence Diagram	9
4 - System Design	10
4.1 - Block Diagram	10
4.2 - Pressure Sensor Driver State Machine	11
4.3 - Alarm Actuator Driver State Machine	12
4.4 - Alarm Monitor State Machine	13
4.5 - Main Algorithm State Machine	14
5 - Source Code	15
5.1 - Alarm Controller	15
5.2 - Pressure Controller	16
5.3 - Pressure Sensor	17
5.4 - Main	18
5.5 - Driver	19
5.6 - States	20
5.7 - Makefile	20
5.8 - Startup File	21
5.9 - Linker Script File	22
5.10 - Symbols File	23
5.11 - Map File	24
6 - Simulation Results	26
6.1 - Ttool Result	26
6.2 - Proteus Result	27

Table Of Figure

Figure 1 : Requirement Diagram.....	4
Figure 2 : Agile Methodology.....	5
Figure 3 : Use Case Diagram.....	6
Figure 4 : Activity Diagram.....	7
Figure 5 : Sequence Diagram.....	8
Figure 6 : Block Diagram.....	9
Figure 7 : Pressure Sensor Driver State Machine.....	10
Figure 8 : Alarm Actuator Driver State Machine.....	11
Figure 9 : Alarm Monitor State Machine.....	12
Figure 10 : Main Algorithm State Machine.....	13

1 - Client Requirement

A "client" expects to deliver the software of the following system specification (from the client) A pressure controller informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin the alarm duration equals 60 seconds.

1.1 - Requirement Diagram

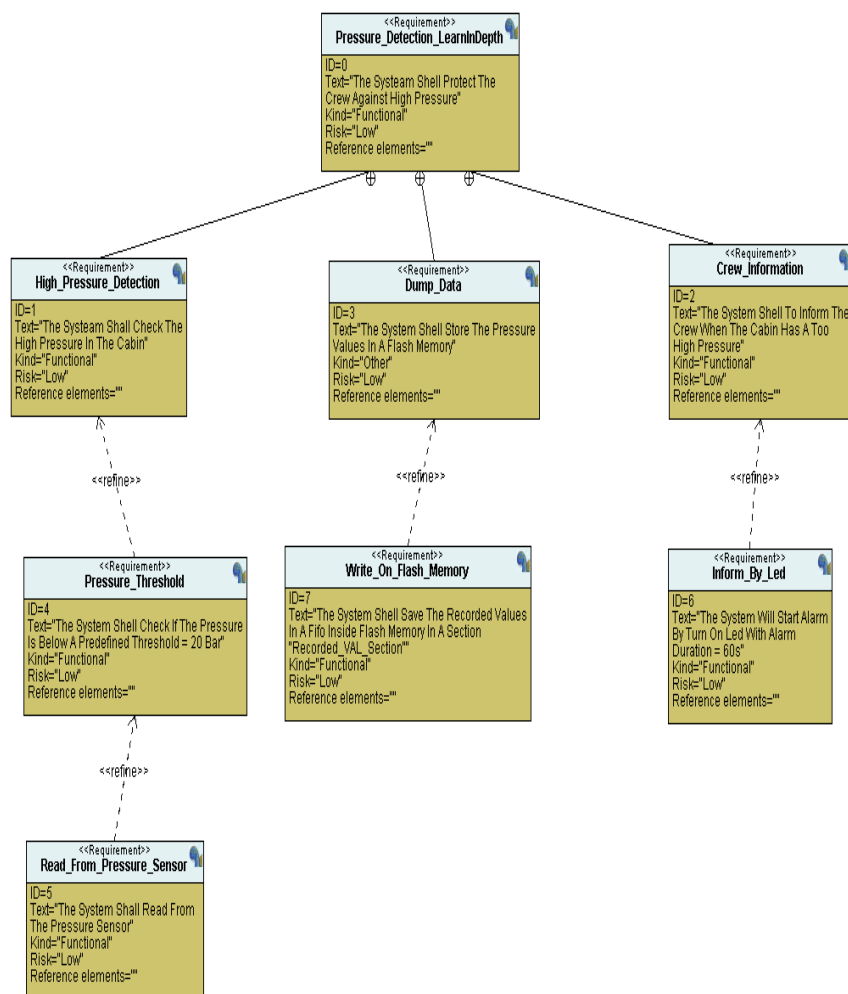


Figure 1 : Requirement Diagram

2 - Methodology

Agile is a flexible and collaborative project management approach that prioritizes iterative development, adaptability to change, and continuous improvement. It fosters regular communication among team members and stakeholders, delivering small, functional increments to meet evolving requirements efficiently. Agile is not just a methodology but a mindset that values individuals, interactions, and the delivery of working solutions in dynamic environments.

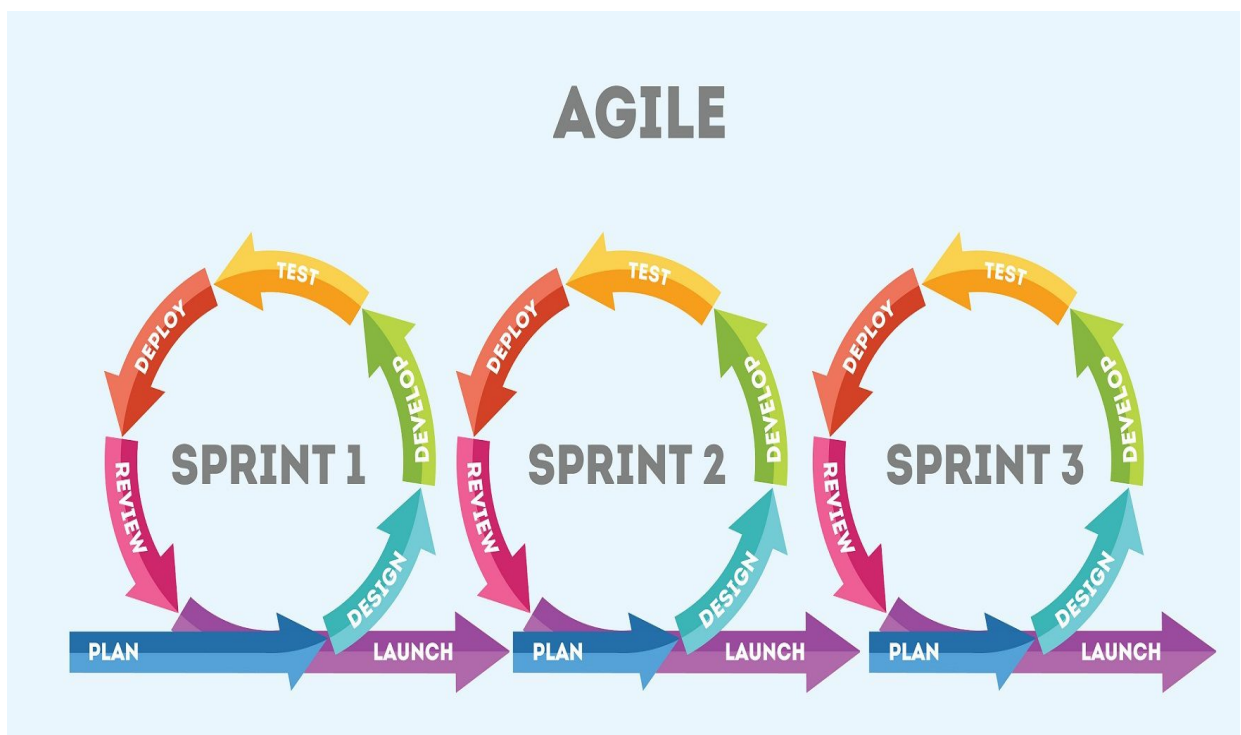


Figure 2 : Agile Methodology

3 - System Analysis

3.1 - Use Case Diagram

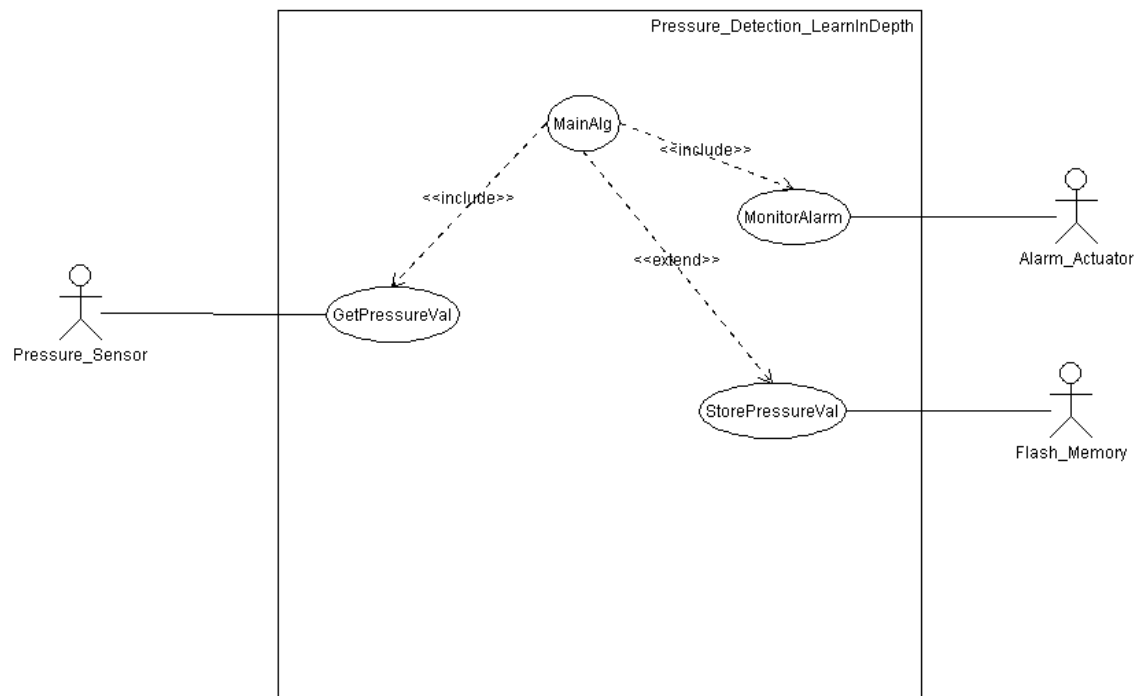


Figure 3 : Use Case Diagram

3.2 - Activity Diagram

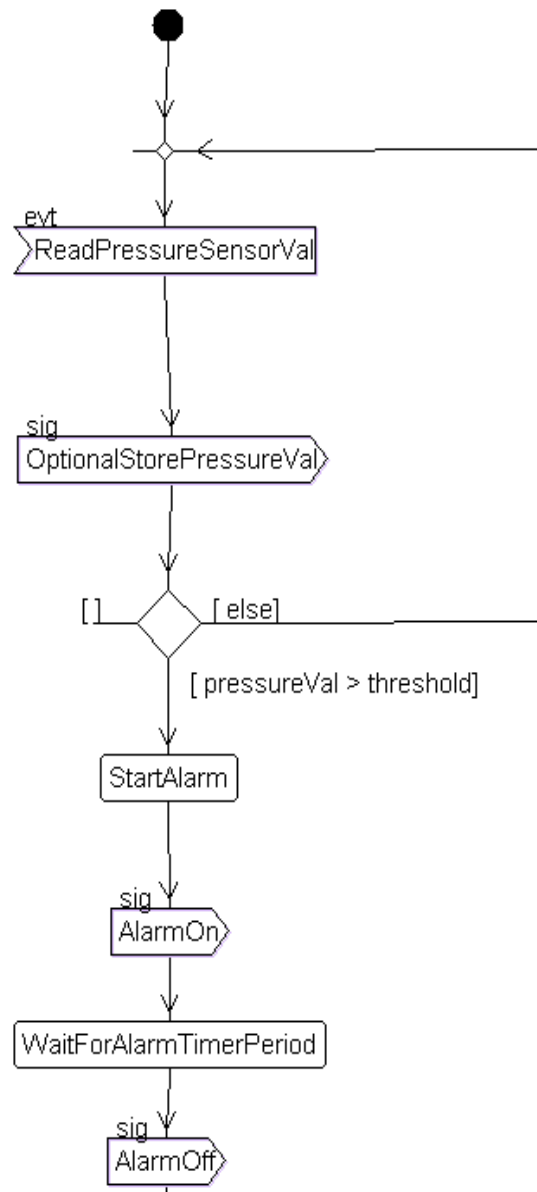


Figure 4 : Activity Diagram

3.3 - Sequence Diagram

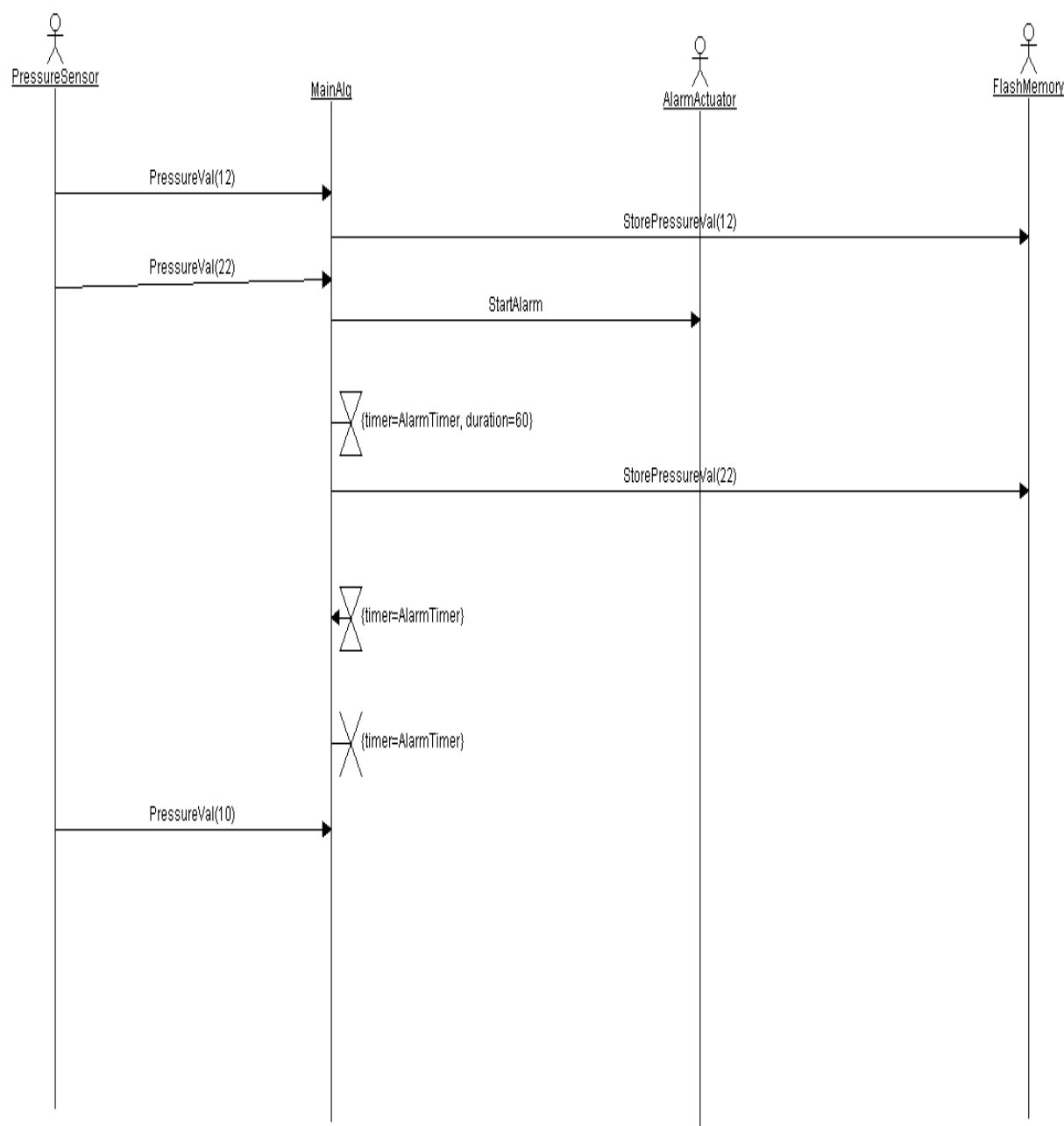


Figure 5 : Sequence Diagram

4 - System Design

4.1 - Block Diagram

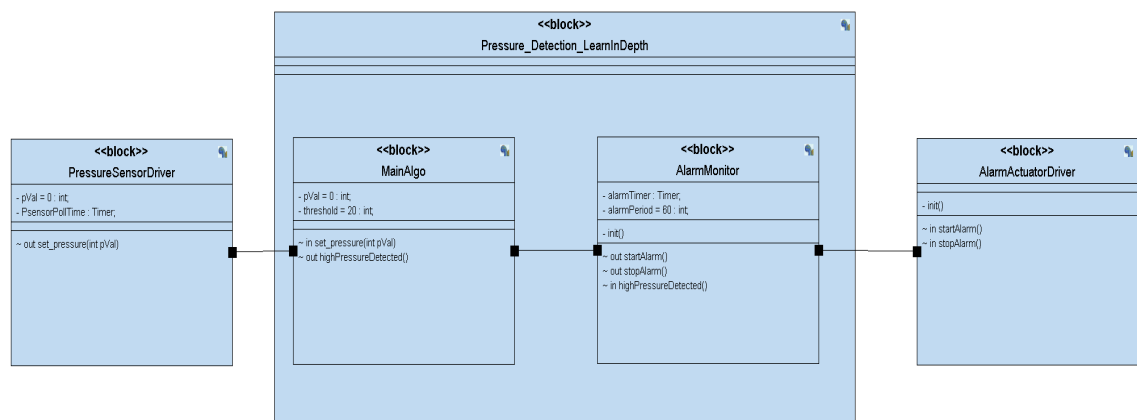


Figure 6 : Block Diagram

4.2 - Pressure Sensor Driver State Machine

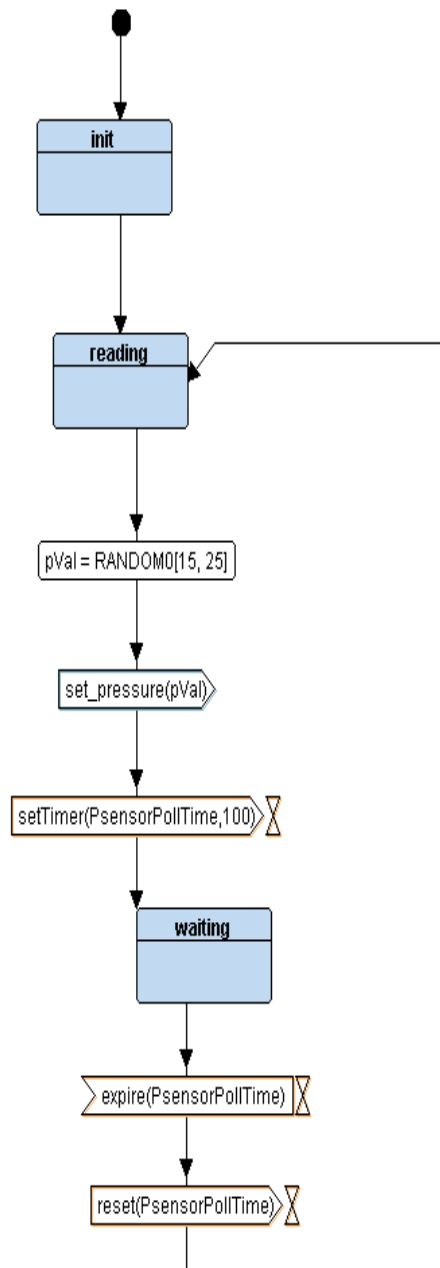


Figure 7 : Pressure Sensor Driver State Machine

4.3 - Alarm Actuator Driver State Machine

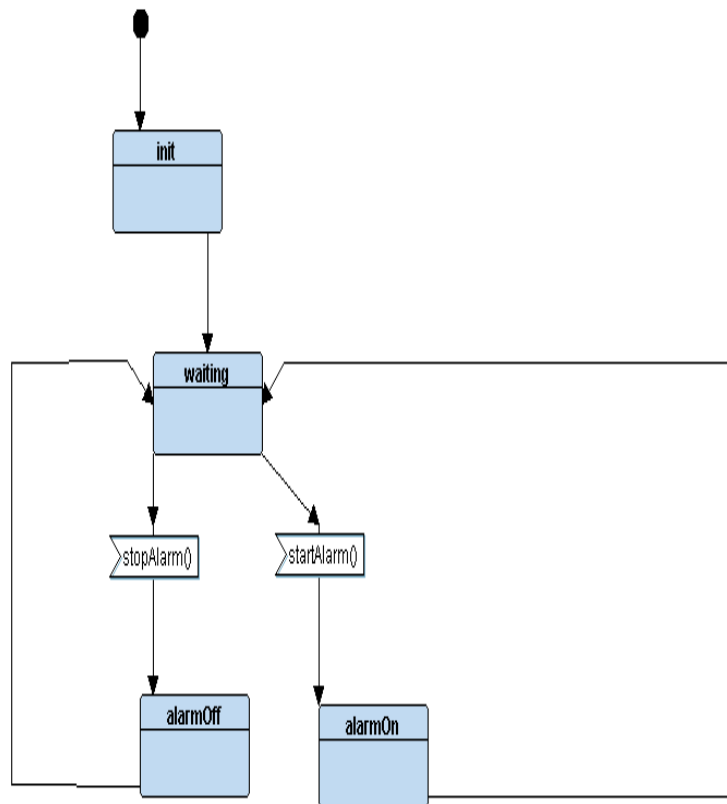


Figure 8 : Alarm Actuator Driver State Machine

4.4 - Alarm Monitor State Machine

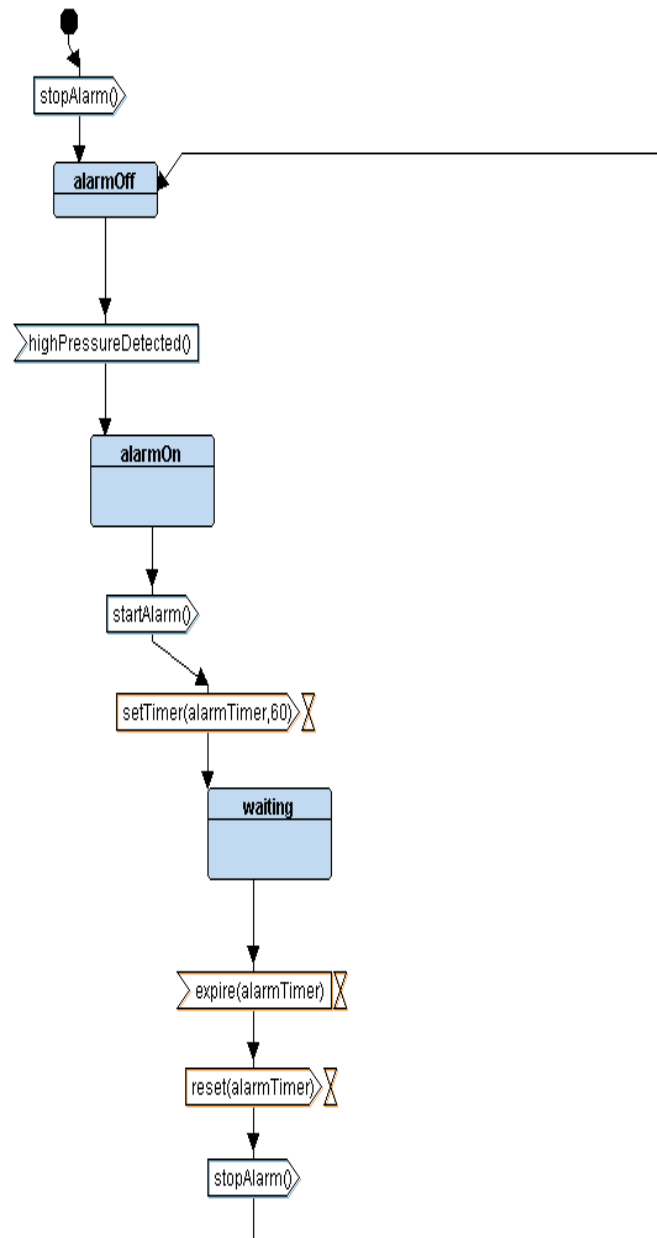


Figure 9 : Alarm Monitor State Machine

4.5 - Main Algorithm State Machine

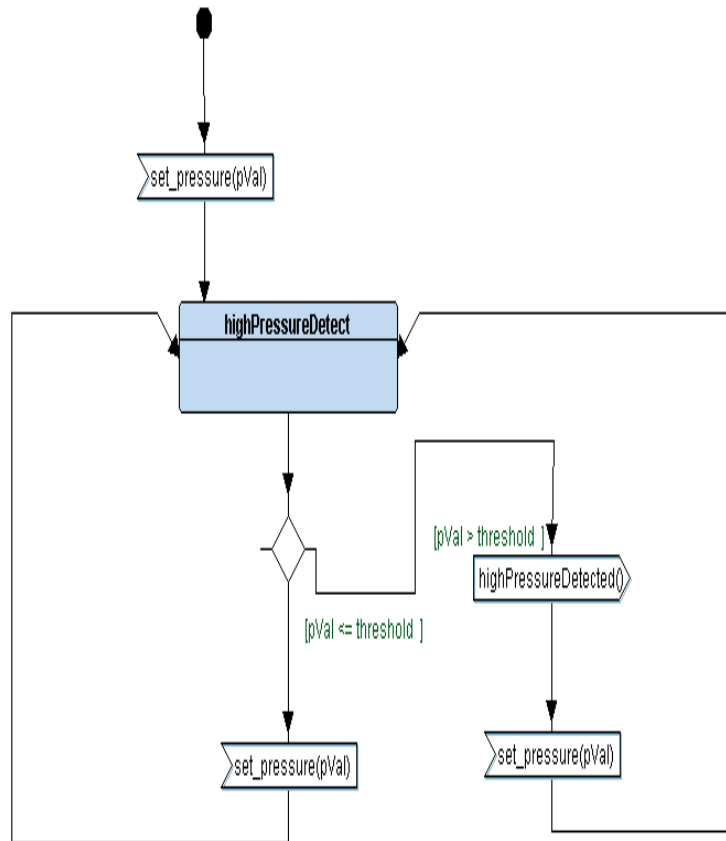


Figure 10 : Main Algorithm State Machine

5 - Source Code

5.1 - Alarm Controller

```
9
10 #include "AlarmController.h"
11 #include "driver.h"
12
13 static EN_AC_States_t ACCurrentState;
14 void (*AC_State)();
15
16 void StartAlarm()
17 {
18     ACCurrentState = AC_ALARM_ON; // set current state
19     AC_State = STATE(AC_ALARM_ON); // set the new state
20     AC_State();
21 }
22 void StopAlarm()
23 {
24     ACCurrentState = AC_ALARM_OFF; // set current state
25     AC_State = STATE(AC_ALARM_OFF); // set the new state
26     AC_State();
27 }
28
29 STATE_DEFINE(AC_ALARM_OFF)
30 {
31     Set_Alarm_actuator(AC_ALARM_OFF); // set the alarm to off
32     AC_State = STATE(AC_WAITING); // set new state
33 }
34
35 STATE_DEFINE(AC_ALARM_ON)
36 {
37     Set_Alarm_actuator(AC_ALARM_ON); // set the alarm to on
38     AC_State = STATE(AC_WAITING); // set new state
39 }
40
41 STATE_DEFINE(AC_WAITING)
42 {
43     ACCurrentState = AC_WAITING; //set current state
44     AC_State = STATE(AC_WAITING); //set the new state in the pointer
45 }
```

```
1  /*****
2  // Author      : Sherif Ashraf Khadr
3  // Project     : 00_Pressure_Detection
4  // File        : AlarmController.h
5  // Date        : Dec 7, 2023
6  // GitHub      : https://github.com/sherifkhadr
7  *****/
8
9  #ifndef ALARMCONTROLLER_H_
10 #define ALARMCONTROLLER_H_
11
12 #include "states.h"
13
14 typedef enum
15 {
16     AC_ALARM_ON = 0,
17     AC_ALARM_OFF,
18     AC_WAITING
19 }EN_AC_States_t;
20
21
22 STATE_DEFINE(AC_ALARM_OFF);
23 STATE_DEFINE(AC_ALARM_ON);
24 STATE_DEFINE(AC_WAITING);
25
26 extern void (*AC_State)();
27
28 #endif /* ALARMCONTROLLER_H_ */
```

5.2 - Pressure Controller

```
1  /**
2  // Author      : Sherif Ashraf Khadr
3  // Project     : 00_Pressure_Detection
4  // File        : PressureController.c
5  // Date        : Dec 7, 2023
6  // GitHub      : https://github.com/sherifkhadr
7  /**
8
9  #include "PressureController.h"
10 #include "PressureSensor.h"
11 #include "AlarmController.h"
12 #include "driver.h"
13
14 #define THRESHOLD_VAL      20
15 static int currentPressure = 0;
16 void (*PC_State)();
17
18 void highPressureDetected()
19 {
20     PC_State = STATE(PC_HIGH_PRESSURE);    // set the state
21 }
22
23 void setPressureVal(int val)
24 {
25     currentPressure = val; // get the value of the pressure
26 }
27
28 void PC_init(void)
29 {
30     PS_init();
31     AC_State = STATE(AC_ALARM_OFF);
32     PS_State = STATE(PS_READING);
33     PC_State = STATE(PC_BELOW_PRESSURE);
34     AC_State();
35     PC_State();
36 }
37
38 void PC_pressureMonitor(void)
39 {
40     while(1)
41     {
42         PS_State();
43         if(currentPressure >= THRESHOLD_VAL)
44             highPressureDetected();
45         PC_State();
46     }
47 }
48
49 STATE_DEFINE(PC_HIGH_PRESSURE)
50 {
51     StartAlarm();// Start alarm
52     Delay(6000); // Delay for 60s
53     StopAlarm(); // Stop alarm
54     PC_State = STATE(PC_BELOW_PRESSURE); // set the state of sensor module to PC_BELOW_PRESSURE
55 }
56
57 STATE_DEFINE(PC_BELOW_PRESSURE)
58 {
59     /*Do Nothing*/
60 }
```

```

1  /**
2  // Author      : Sherif Ashraf Khadr
3  // Project     : 00_Pressure_Detection
4  // File        : PressureController.h
5  // Date        : Dec 7, 2023
6  // GitHub      : https://github.com/sherifkhadr
7  */
8
9
10 #ifndef PRESSURECONTROLLER_H_
11 #define PRESSURECONTROLLER_H_
12
13 #include "states.h"
14
15
16 typedef enum
17 {
18     PC_CHECK_PRESSURE = 0,
19     PC_HIGH_PRESSURE,
20     PC_BELOW_PRESSURE
21 }EN_PC_States_t;
22
23 STATE_DEFINE(PC_HIGH_PRESSURE);
24 STATE_DEFINE(PC_BELOW_PRESSURE);
25
26
27 extern void (*PC_State)();
28
29 void PC_init(void);
30 void PC_pressureMonitor(void);
31
32 #endif /* PRESSURECONTROLLER_H_ */

```

5.3 - Pressure Sensor

```

9  #include "PressureSensor.h"
10 #include "driver.h"
11
12 static int pressureVal = 0;
13 static EN_PS_States_t PSCurrentState;
14
15 void (*PS_State)();
16
17 void PS_init(void)
18 {
19     GPIO_INITIALIZATION(); // init sensor
20     PS_State = STATE(PS_READING); // set the new state in the pointer
21 }
22
23 STATE_DEFINE(PS_READING)
24 {
25     PSCurrentState = PS_READING; // set current state
26     pressureVal = getPressureVal(); // read the value of the sensor
27     PS_State = STATE(PS_SEND_VAL); // set the new state in the pointer
28 }
29
30 STATE_DEFINE(PS_SEND_VAL)
31 {
32     PSCurrentState = PS_SEND_VAL; // set current state
33     setPressureVal(pressureVal); // update the value
34     PS_State = STATE(PS_WAITING); // set the new state in the pointer
35 }
36
37 STATE_DEFINE(PS_WAITING)
38 {
39     PSCurrentState = PS_WAITING; // set current state
40     Delay(10000); // wait for the delay
41     PS_State = STATE(PS_READING); // set the new state in the pointer
42 }

```



```

1  /*****
2  // Author      : Sherif Ashraf Khadr
3  // Project     : 00_Pressure_Detection
4  // File        : PressureSensor.h
5  // Date        : Dec 7, 2023
6  // GitHub      : https://github.com/sherifkhadr
7  *****/
8
9  #ifndef PRESSURESENSOR_H_
10 #define PRESSURESENSOR_H_
11
12 #include "states.h"
13
14
15 typedef enum
16 {
17     PS_READING = 0,
18     PS_SEND_VAL,
19     PS_WAITING
20 }EN_PS_States_t;
21
22 void PS_init(void);
23
24 STATE_DEFINE(PS_READING);
25 STATE_DEFINE(PS_SEND_VAL);
26 STATE_DEFINE(PS_WAITING);
27
28 extern void (*PS_State)();
29
30 #endif /* PRESSURESENSOR_H_ */

```

5.4 - Main

```

1  /*****
2  // Author      : Sherif Ashraf Khadr
3  // Project     : 00_Pressure_Detection
4  // File        : main.c
5  // Date        : Dec 7, 2023
6  // GitHub      : https://github.com/sherifkhadr
7  *****/
8
9
10 #include "PressureController.h"
11
12
13 int main(void)
14 {
15
16     PC_init();
17     PC_pressureMonitor();
18     return 0;
19 }

```

5.5 - Driver

```
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 &= 0xFF0FFFFFFF;
25     GPIOA_CRL |= 0x00000000;
26     GPIOA_CRH &= 0xFF0FFFFFFF;
27     GPIOA_CRH |= 0x22222222;
28 }
29
```

```
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
```

5.6 - States

```
1  /*******  
2  // Author      : Sherif Ashraf Khadr  
3  // Project     : 00_Pressure_Detection  
4  // File        : states.h  
5  // Date        : Dec 7, 2023  
6  // GitHub      : https://github.com/sherifkhadr  
7  /*******  
8  
9  #ifndef STATES_H_  
10 #define STATES_H_  
11  
12 #include <stdio.h>  
13 #include <stdlib.h>  
14  
15 //Automatic STATE Function genrated  
16 #define STATE_DEFINE(_stateFun_) void ST_##_stateFun_()  
17 #define STATE(_stateFun_) ST_##_stateFun_  
18  
19 // Signals  
20 void setPressureVal(int val);  
21 void highPressureDetected();  
22 void StartAlarm();  
23 void StopAlarm();  
24  
25 #endif /* STATES_H_ */  
26
```

5.7 - Makefile

```
1  #/*******  
2  #// Author      : Sherif Ashraf Khadr  
3  #// Project     : 00_Pressure_Detection  
4  #// File        : Makefile  
5  #// Date        : Dec 7, 2023  
6  #// GitHub      : https://github.com/sherifkhadr  
7  #/*******  
8  
9  
10 CC=arm-none-eabi-  
11 CFLAGS=-mcpu=cortex-m3 -gdwarf-2  
12 INCS=-I .  
13 LIBS=  
14 SRC = $(wildcard *.c)  
15 OBJ = $(SRC:.c=.o)  
16 AS = $(wildcard *.s)  
17 ASOBJ = $(AS:.s=.o)  
18 projectName=Pressure_Detection  
19  
20 all: $(projectName).bin  
21     @echo "Build Is Done"  
22  
23 %.o: %.c  
24     $(CC)gcc.exe -c $(INCS) $(CFLAGS) $< -o $@  
25  
26 $(projectName).elf: $(OBJ) $(ASOBJ)  
27     $(CC)ld.exe -T linkerscript.ld $(LIBS) $(OBJ) $(ASOBJ) -o $(projectName).elf -Map=$(projectName).map  
28  
29 $(projectName).bin: $(projectName).elf  
30     $(CC)objcopy.exe -O binary $< $@  
31  
32 clean_all:  
33     rm *.o *.elf *.bin  
34  
35 clean:  
36     rm *.elf *.bin
```

5.8 - Startup File

```
10  #include <stdint.h>
11
12  extern int main(void);
13  extern unsigned int _E_TEXT;
14  extern unsigned int _S_DATA;
15  extern unsigned int _E_DATA;
16  extern unsigned int _S_BSS;
17  extern unsigned int _E_BSS;
18  extern unsigned int _STACK_TOP;
19
20  void Default_Handler(void);
21  void Reset_Handler(void);
22  void NMI_Handler(void) __attribute__((weak, alias("Default_Handler")));
23  void H_Fault_Handler(void) __attribute__((weak, alias("Default_Handler")));
24  void MM_Fault_Handler(void) __attribute__((weak, alias("Default_Handler")));
25  void Bus_Fault_Handler(void) __attribute__((weak, alias("Default_Handler")));
26  void Usage_Fault_Handler(void) __attribute__((weak, alias("Default_Handler")));
27
28  uint32_t vectors[] __attribute__((section(".vectors"))) =
29  {
30      (uint32_t) &_STACK_TOP,
31      (uint32_t) &Reset_Handler,
32      (uint32_t) &NMI_Handler,
33      (uint32_t) &H_Fault_Handler,
34      (uint32_t) &MM_Fault_Handler,
35      (uint32_t) &Bus_Fault_Handler,
36      (uint32_t) &Usage_Fault_Handler,
37  };
38
39
40  void Default_Handler(void)
41  {
42      Reset_Handler();
43  }
```

```
45  void Reset_Handler(void)
46  {
47      /* copy data section from flash to sram */
48
49      unsigned int DATA_SIZE = (unsigned char *)&_E_DATA - (unsigned char *)&_S_DATA ;
50      unsigned char* P_src = (unsigned char *)&_E_TEXT ;
51      unsigned char* P_dst = (unsigned char *)&_S_DATA ;
52
53      for(int i=0 ; i<DATA_SIZE ; i++)
54      {
55          *((unsigned char *)P_dst++) = *((unsigned char *)P_src++);
56      }
57
58      /* init .bss section in sram = 0 */
59
60      unsigned int BSS_SIZE = (unsigned char *)&_E_BSS - (unsigned char *)&_S_BSS ;
61      P_dst = (unsigned char *)&_S_BSS;
62
63      for(int i=0 ; i<BSS_SIZE ; i++)
64      {
65          *((unsigned char *)P_dst++) = (unsigned char)0;
66      }
67
68      /* jump to the main */
69
70      main();
71  }
```

5.9 - Linker Script File

```
9
10 MEMORY
11 {
12     flash(RX) : ORIGIN = 0x08000000 , LENGTH = 128K
13     sram(RWX) : ORIGIN = 0x20000000 , LENGTH = 20K
14 }
15
16 SECTIONS
17 {
18
19     .text : {
20         *(.vectors*)
21         *(.text*)
22         *(.rodata*)
23         _E_TEXT = . ;
24     }> flash
25
26     .data : {
27         _S_DATA = . ;
28         *(.data*)
29         . = ALIGN(4) ;
30         _E_DATA = . ;
31     }> sram AT> flash
32
33     .bss : {
34         _S_BSS = . ;
35         *(.bss*)
36         _E_BSS = . ;
37         . = ALIGN(4) ;
38         . = . + 0x1000 ;
39         _STACK_TOP = . ;
40     }> sram
41 }
```

5.10 - Symbols File

```
1 arm-none-eabi-nm.exe .\Pressure_Detection.elf
2
3 2000000c B _E_BSS
4 20000000 D _E_DATA
5 080003b0 T _E_TEXT
6 20000000 B _S_BSS
7 20000000 D _S_DATA
8 2000100c B _STACK_TOP
9 20001010 B AC_State
10 20000005 b ACCurrentState
11 0800001c W Bus_Fault_Handler
12 20000008 b currentPressure
13 0800001c T Default_Handler
14 080002ec T Delay
15 0800030c T getPressureVal
16 08000360 T GPIO_INITIALIZATION
17 0800001c W H_Fault_Handler
18 08000210 T highPressureDetected
19 080000ac T main
20 0800001c W MM_Fault_Handler
21 0800001c W NMI_Handler
22 08000248 T PC_init
23 0800028c T PC_pressureMonitor
24 20001014 B PC_State
25 20000000 b pressureVal
26 080000c0 T PS_init
27 2000100c B PS_State
28 20000004 b PSCurrentState
29 08000028 T Reset_Handler
30 08000324 T Set_Alarm_actuator
31 0800022c T setPressureVal
32 080001b4 T ST_AC_ALARM_OFF
33 080001d0 T ST_AC_ALARM_ON
34 080001ec T ST_AC_WAITING
35 080002e0 T ST_PC_BELOW_PRESSURE
36 080002b8 T ST_PC_HIGH_PRESSURE
37 080000dc T ST_PS_READING
38 0800010c T ST_PS_SEND_VAL
39 0800013c T ST_PS_WAITING
40 08000164 T StartAlarm
41 0800018c T StopAlarm
42 0800001c W Usage_Fault_Handler
43 08000000 T vectors
```

5.11 - Map File

```

1  s
2  Allocating common symbols
3  Common symbol      size      file
4
5  PC_State           0x4        PressureController.o
6  PS_State           0x4        PressureSensor.o
7  AC_State           0x4        AlarmController.o
8
9  Memory Configuration
10
11 Name                Origin                Length                Attributes
12 flash               0x08000000           0x00020000           xr
13 sram                 0x20000000           0x00005000           xrw
14 *default*           0x00000000           0xffffffff
15
16 Linker script and memory map
17
18
19 .text               0x08000000           0x3b0
20 *(.vectors*)
21 .vectors            0x08000000           0x1c startup.o
22                      0x08000000           vectors
23 *(.text*)
24 .text              0x0800001c           0x90 startup.o
25                      0x0800001c           Bus_Fault_Handler
26                      0x0800001c           H_Fault_Handler
27                      0x0800001c           MM_Fault_Handler
28                      0x0800001c           Usage_Fault_Handler
29                      0x0800001c           Default_Handler
30                      0x0800001c           NMI_Handler
31                      0x08000028           Reset_Handler
32 .text              0x080000ac           0x12 main.o
33                      0x080000ac           main
34 *fill*              0x080000be           0x2
35 .text              0x080000c0           0xa4 PressureSensor.o
36                      0x080000c0           PS_init
37                      0x080000dc           ST_PS_READING

```

```

38                      0x0800010c           ST_PS_SEND_VAL
39                      0x0800013c           ST_PS_WAITING
40 .text              0x08000164           0xac AlarmController.o
41                      0x08000164           StartAlarm
42                      0x0800018c           StopAlarm
43                      0x080001b4           ST_AC_ALARM_OFF
44                      0x080001d0           ST_AC_ALARM_ON
45                      0x080001ec           ST_AC_WAITING
46 .text              0x08000210           0xdc PressureController.o
47                      0x08000210           highPressureDetected
48                      0x0800022c           setPressureVal
49                      0x08000248           PC_init
50                      0x0800028c           PC_pressureMonitor
51                      0x080002b8           ST_PC_HIGH_PRESSURE
52                      0x080002e0           ST_PC_BELOW_PRESSURE
53 .text              0x080002ec           0xc4 driver.o
54                      0x080002ec           Delay
55                      0x0800030c           getPressureVal
56                      0x08000324           Set_Alarm_actuator
57                      0x08000360           GPIO_INITIALIZATION
58 *(.rodata*)
59                      0x080003b0           _E_TEXT = .
60
61 .glue_7             0x080003b0           0x0
62 .glue_7             0x080003b0           0x0 linker stubs
63
64 .glue_7t            0x080003b0           0x0
65 .glue_7t            0x080003b0           0x0 linker stubs
66
67 .vfp11_veneer       0x080003b0           0x0
68 .vfp11_veneer       0x080003b0           0x0 linker stubs
69
70 .v4_bx              0x080003b0           0x0
71 .v4_bx              0x080003b0           0x0 linker stubs
72
73 .iplt               0x080003b0           0x0

```

```

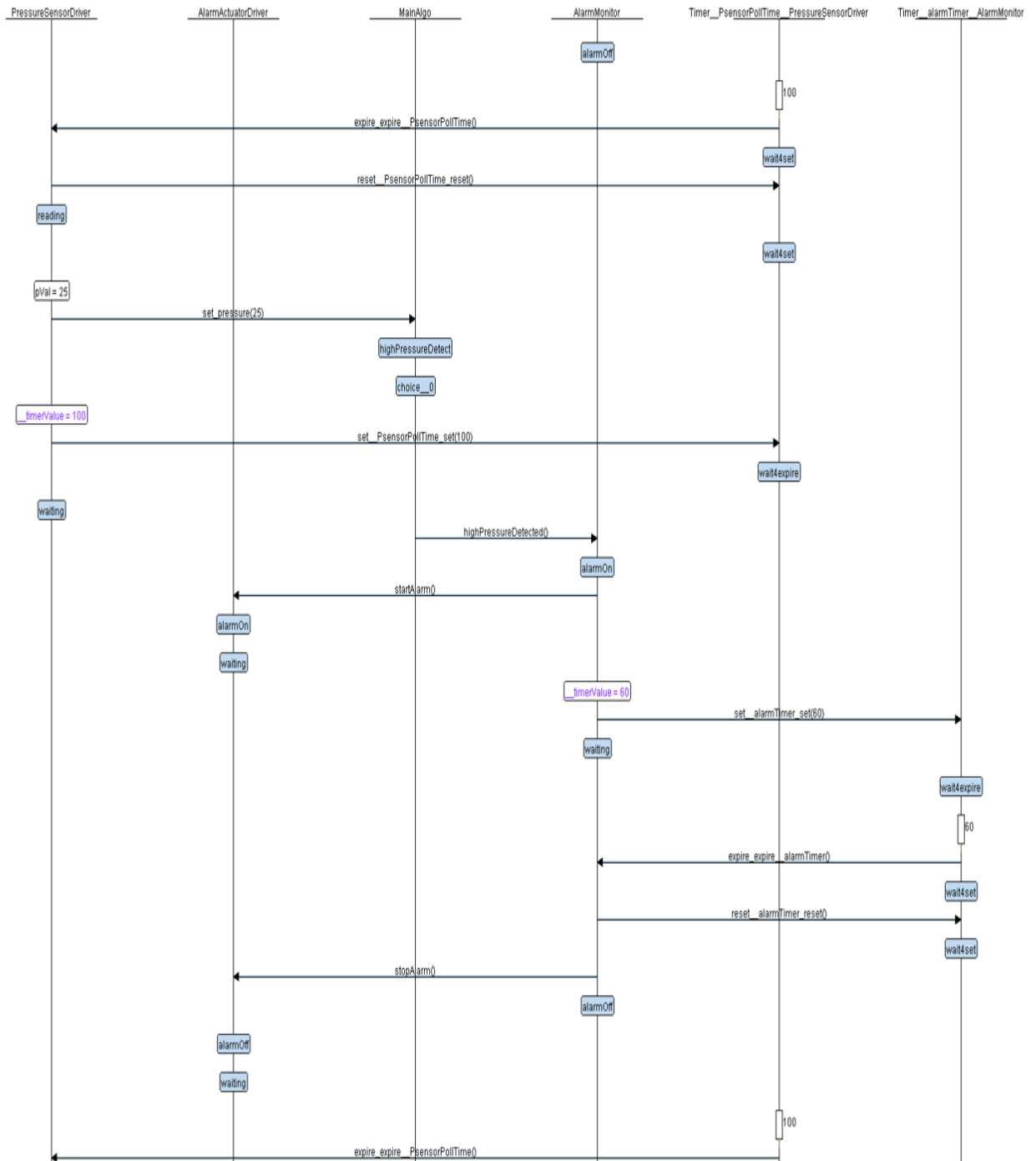
73 ▼ .iplt      0x080003b0      0x0
74   .iplt      0x080003b0      0x0 startup.o
75
76 ▼ .rel.dyn    0x080003b0      0x0
77   .rel.iplt   0x080003b0      0x0 startup.o
78
79 ▼ .data      0x20000000      0x0 load address 0x080003b0
80           0x20000000          _S_DATA = .
81   *(.data*)
82   .data      0x20000000      0x0 startup.o
83   .data      0x20000000      0x0 main.o
84   .data      0x20000000      0x0 PressureSensor.o
85   .data      0x20000000      0x0 AlarmController.o
86   .data      0x20000000      0x0 PressureController.o
87 ▼ .data      0x20000000      0x0 driver.o
88           0x20000000          . = ALIGN (0x4)
89           0x20000000          _E_DATA = .
90
91 ▼ .igot.plt   0x20000000      0x0 load address 0x080003b0
92   .igot.plt   0x20000000      0x0 startup.o
93
94 ▼ .bss       0x20000000      0x1018 load address 0x080003b0
95           0x20000000          _S_BSS = .
96   *(.bss*)
97   .bss       0x20000000      0x0 startup.o
98   .bss       0x20000000      0x0 main.o
99   .bss       0x20000000      0x5 PressureSensor.o
100  .bss       0x20000005      0x1 AlarmController.o
101  *fill*     0x20000006      0x2
102  .bss       0x20000008      0x4 PressureController.o
103 ▼ .bss       0x2000000c      0x0 driver.o
104           0x2000000c          _E_BSS = .
105           0x2000000c          . = ALIGN (0x4)
106           0x2000100c          . = (. + 0x1000)
107 ▼ *fill*     0x2000000c      0x1000
108           0x2000100c          _STACK_TOP = .
109 ▼ COMMON     0x2000100c      0x4 PressureSensor.o

110           0x2000100c          PS_State
111 ▼ COMMON     0x20001010      0x4 AlarmController.o
112           0x20001010          AC_State
113 ▼ COMMON     0x20001014      0x4 PressureController.o
114           0x20001014          PC_State
115 LOAD startup.o
116 LOAD main.o
117 LOAD PressureSensor.o
118 LOAD AlarmController.o
119 LOAD PressureController.o
120 LOAD driver.o
121 OUTPUT(Pressure_Detection.elf elf32-littlearm)
122

```


6 - Simulation Results

6.1 - Ttool Result



6.2 - Proteus Result

