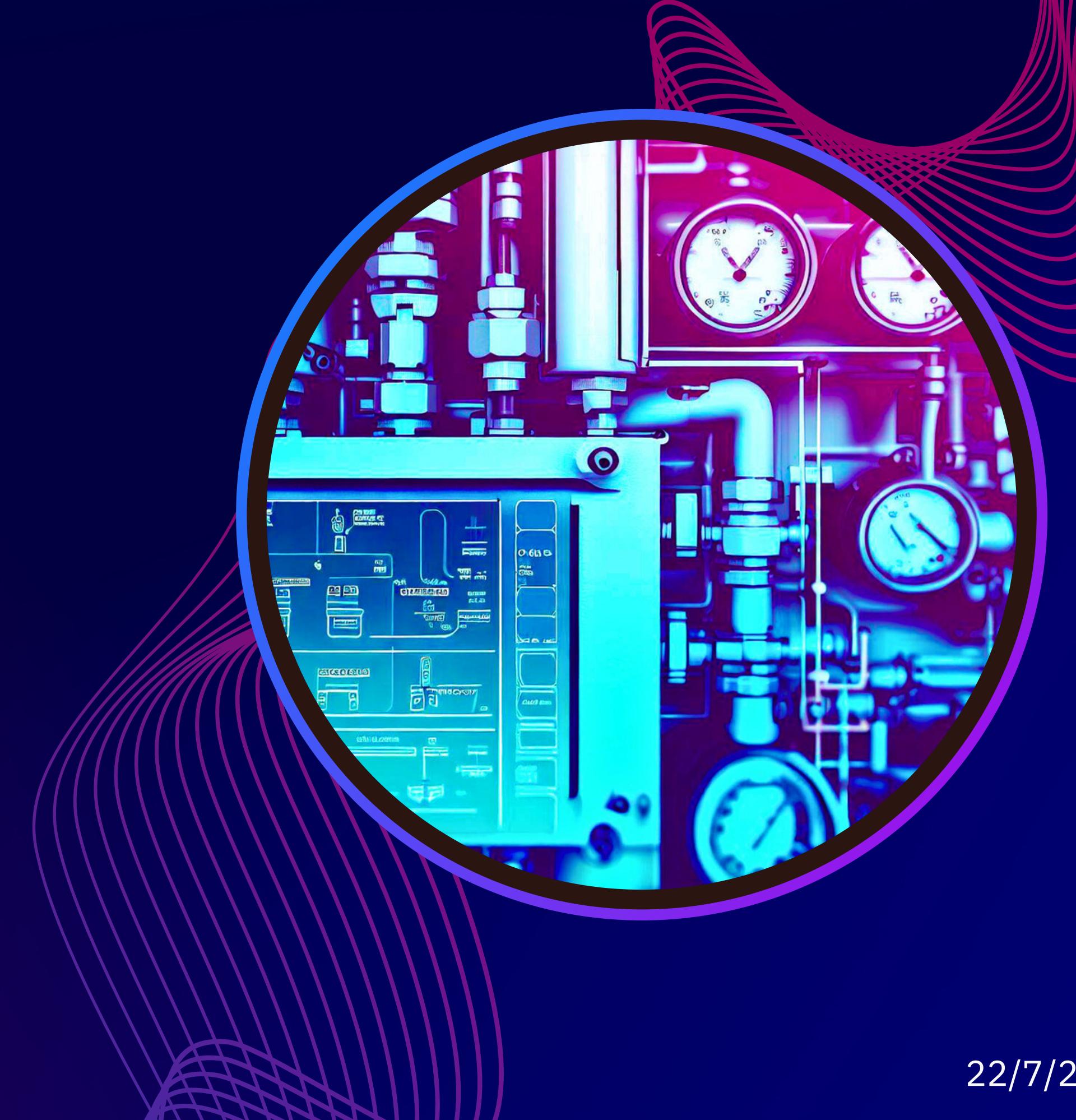


PRESSURE CONTROLLER



PRESENTED BY

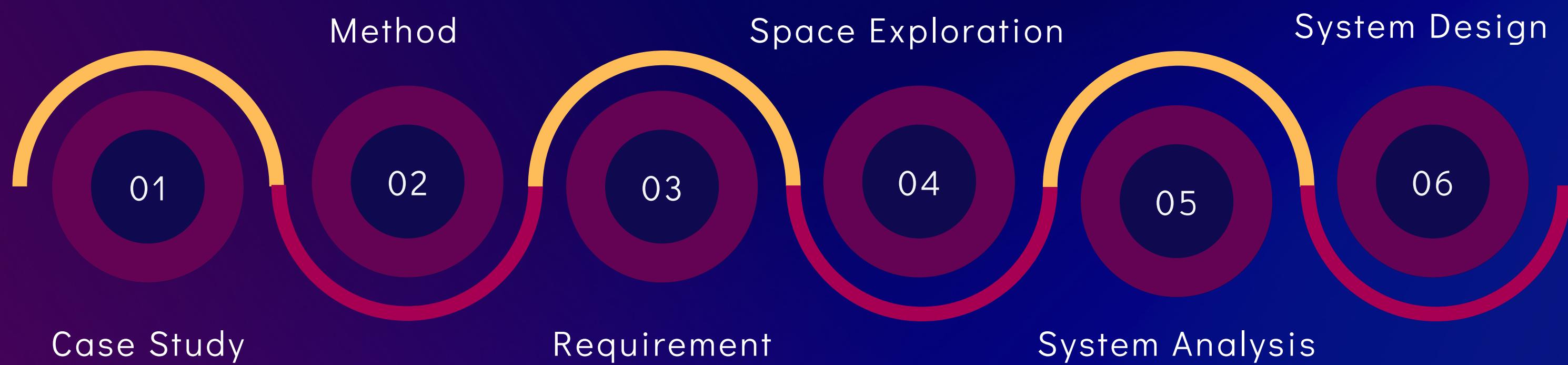
Mohamed Ebrahim Belal

PRESENTED TO

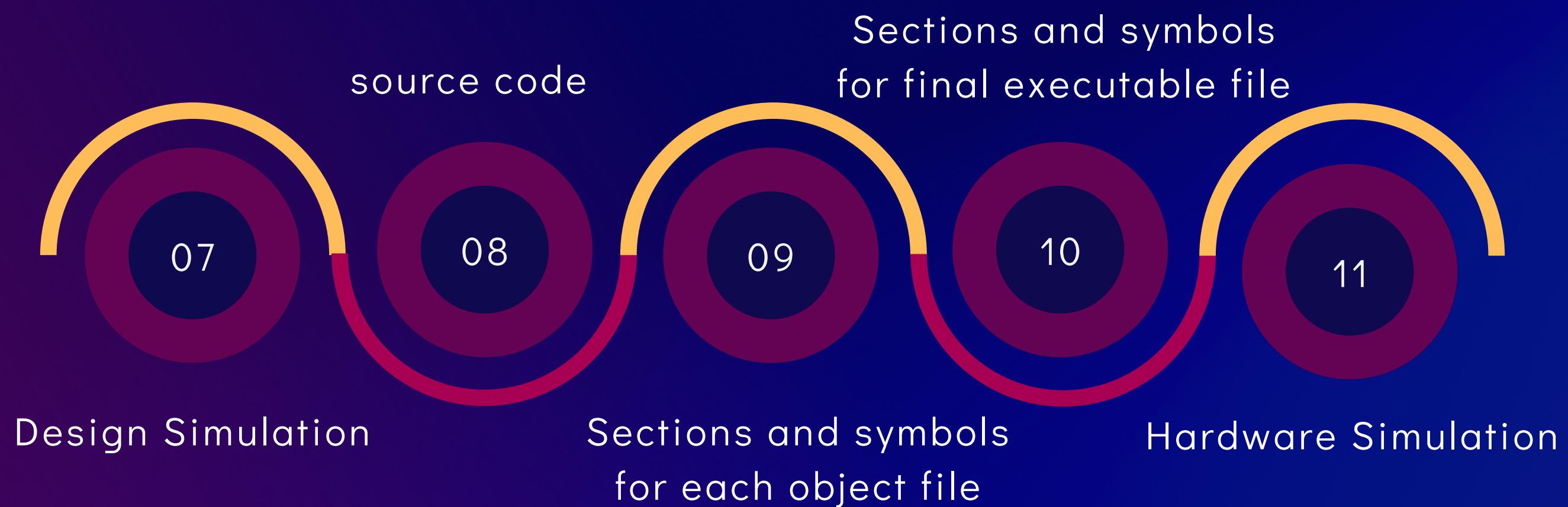
ENG : keroles Shenouda

22/7/2022

System Architecting & Design Sequence



System Architecting & Design Sequence



Case Study

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.
Keep track of the measured values.

Assumptions:

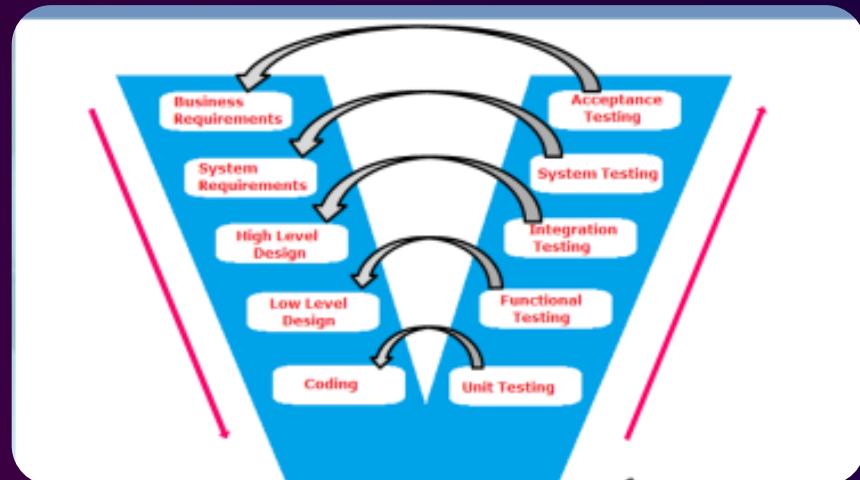
1. The system setup and shutdown procedures are not modeled.
2. The system maintenance is not modeled.
3. The pressure sensor never fails.
4. The alarm never fails.
5. The system never faces power cut.

Methodology

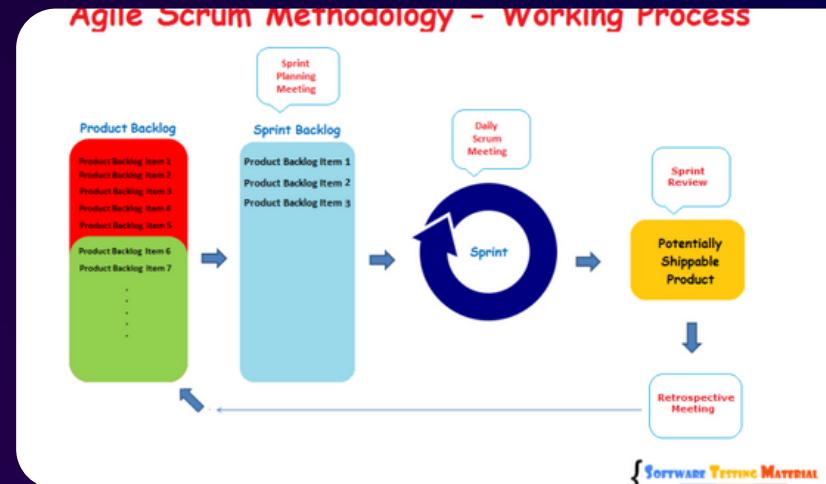
Since the system has multiple modules that are no easy to integrate

There are many methods, such as :

V-model



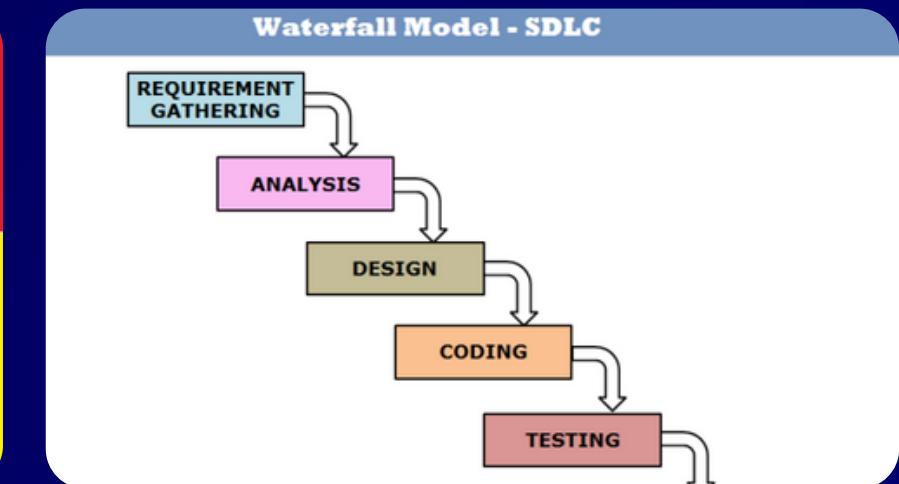
agile model



spiral model

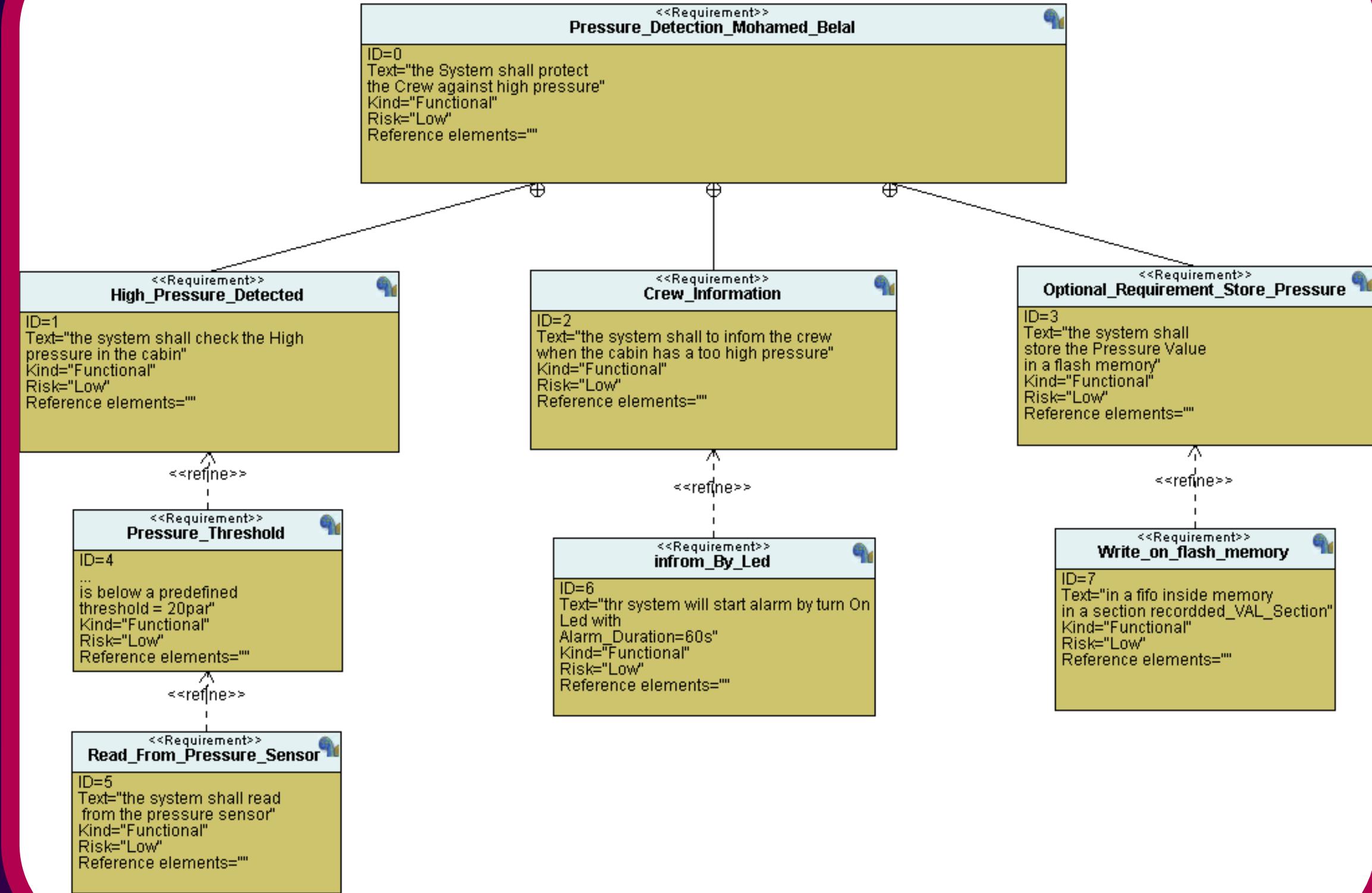


waterfall model



we Will Use V-Model

Requirement



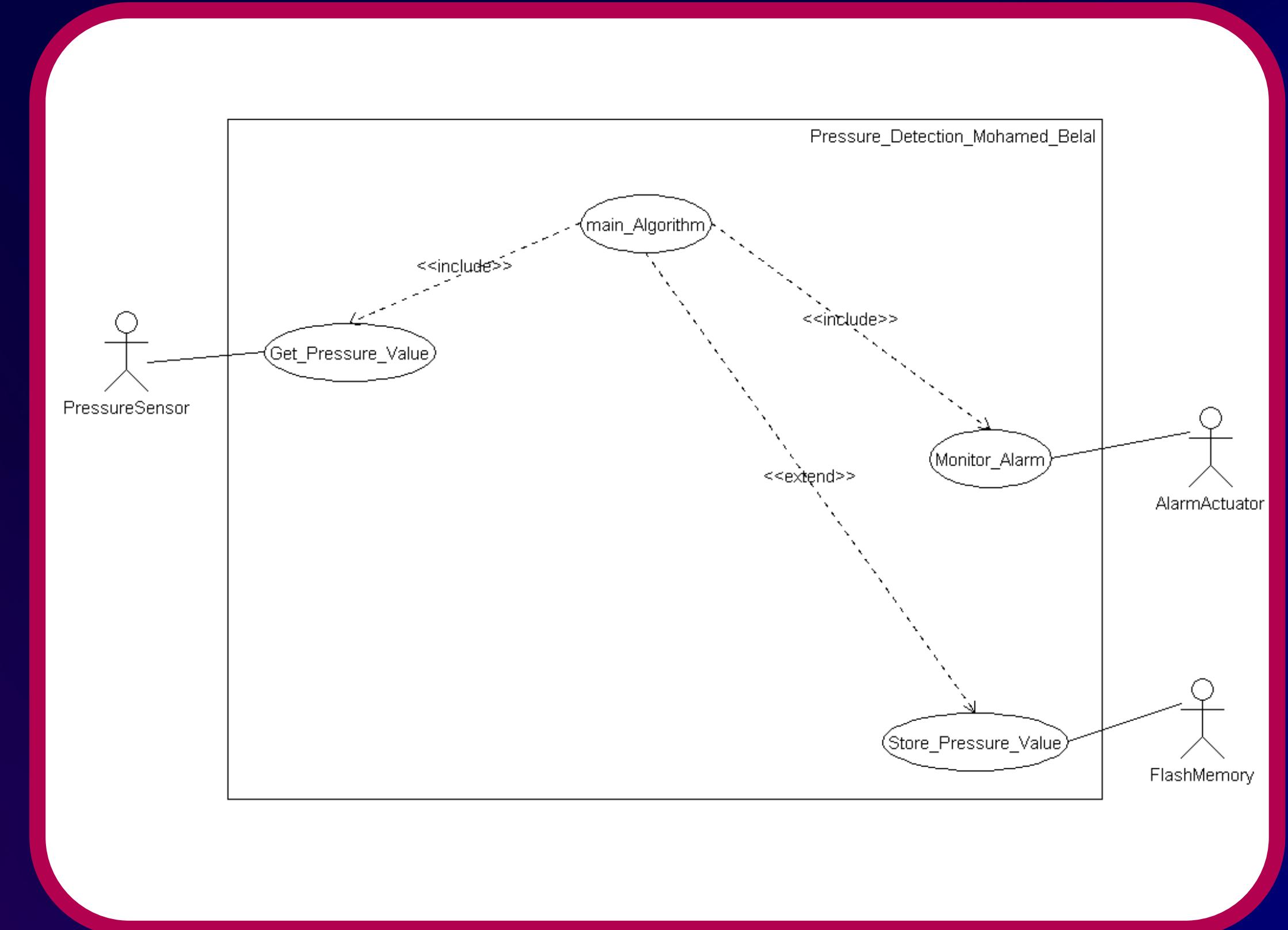
Space Exploration

We Will Use STM32F103C6 microcontroller with a cortex-m3
it's more than enough for this application

System Analysis



Use Case Diagram

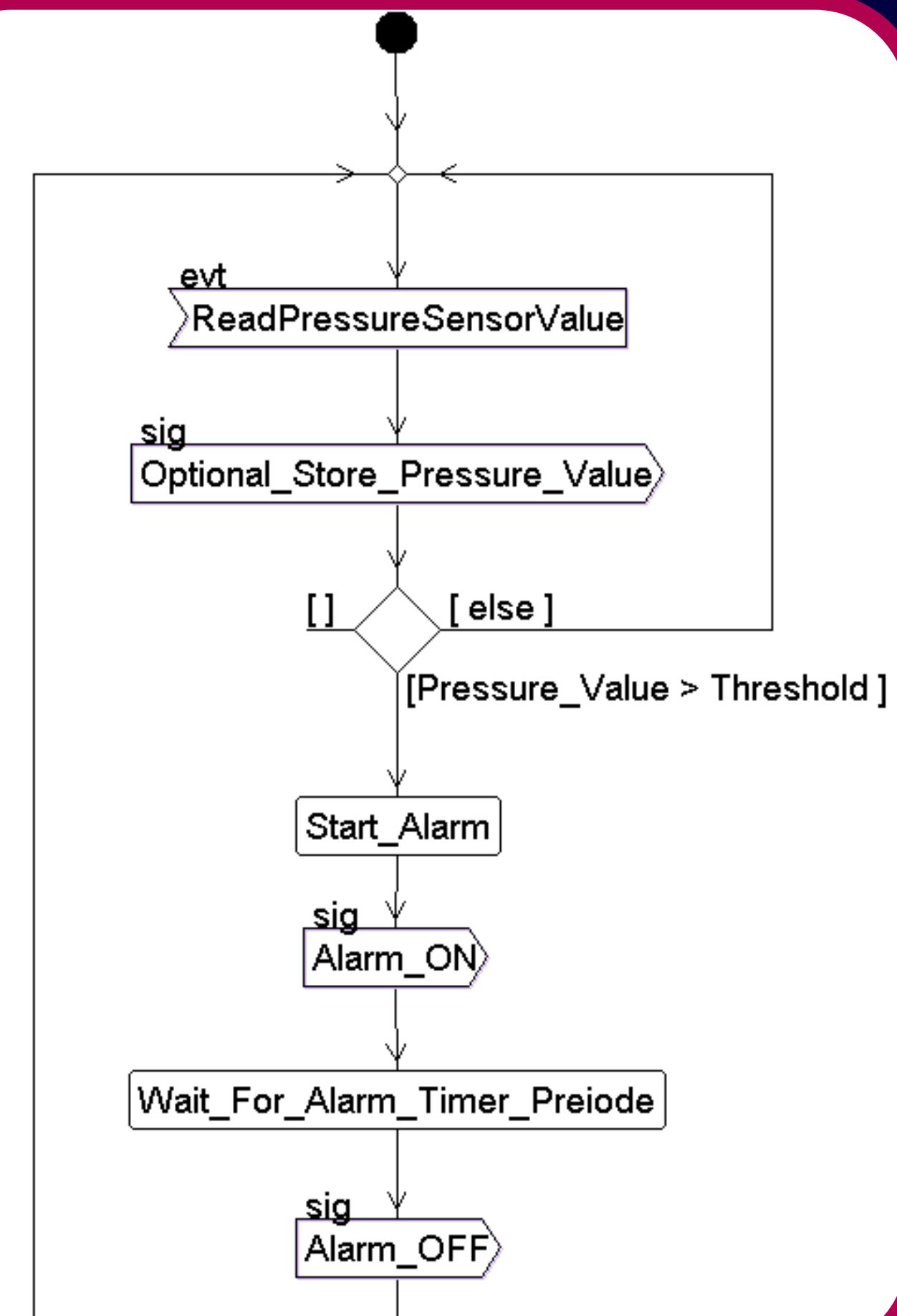


05

System Analysis

2

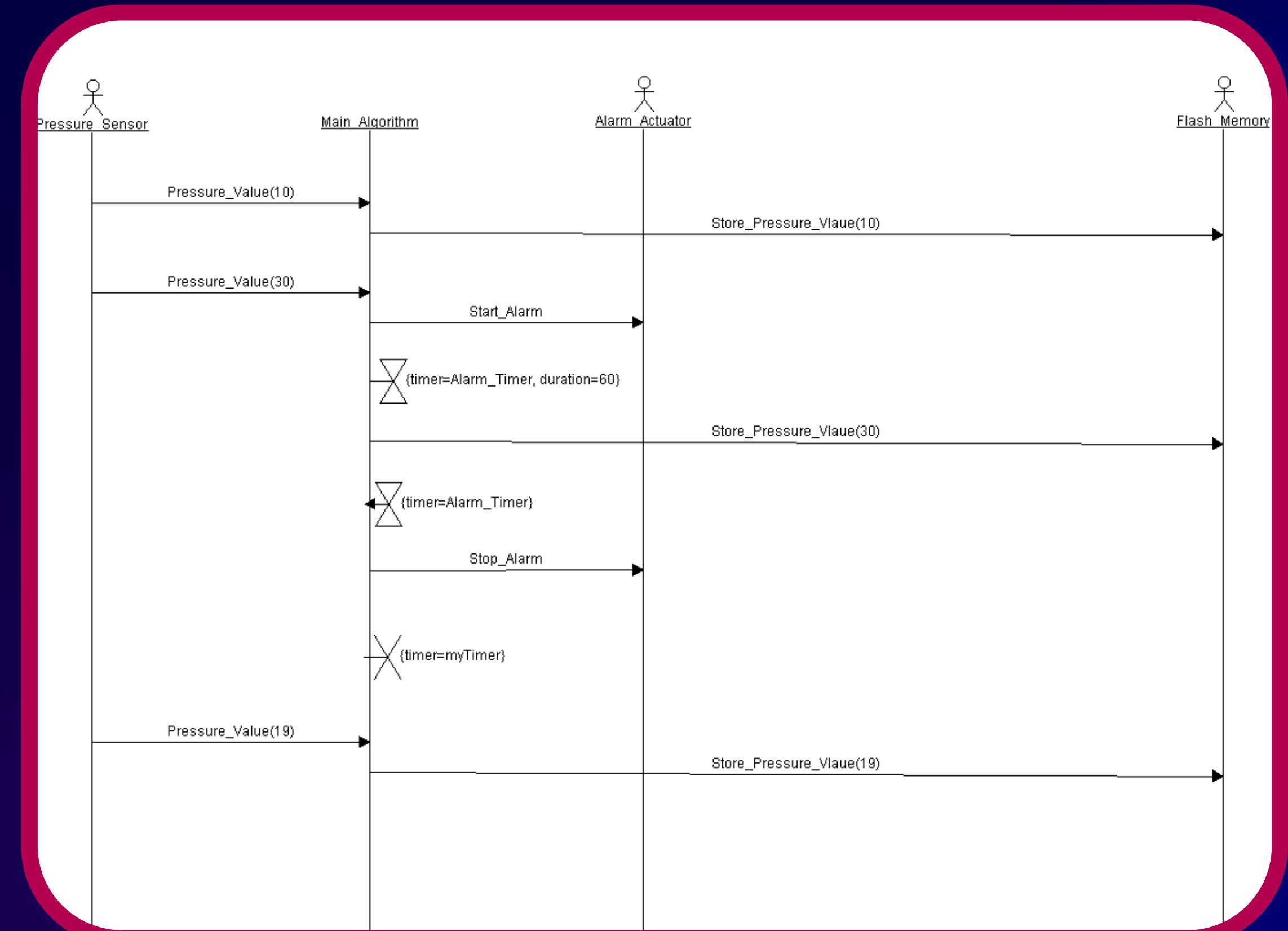
Activity Diagram



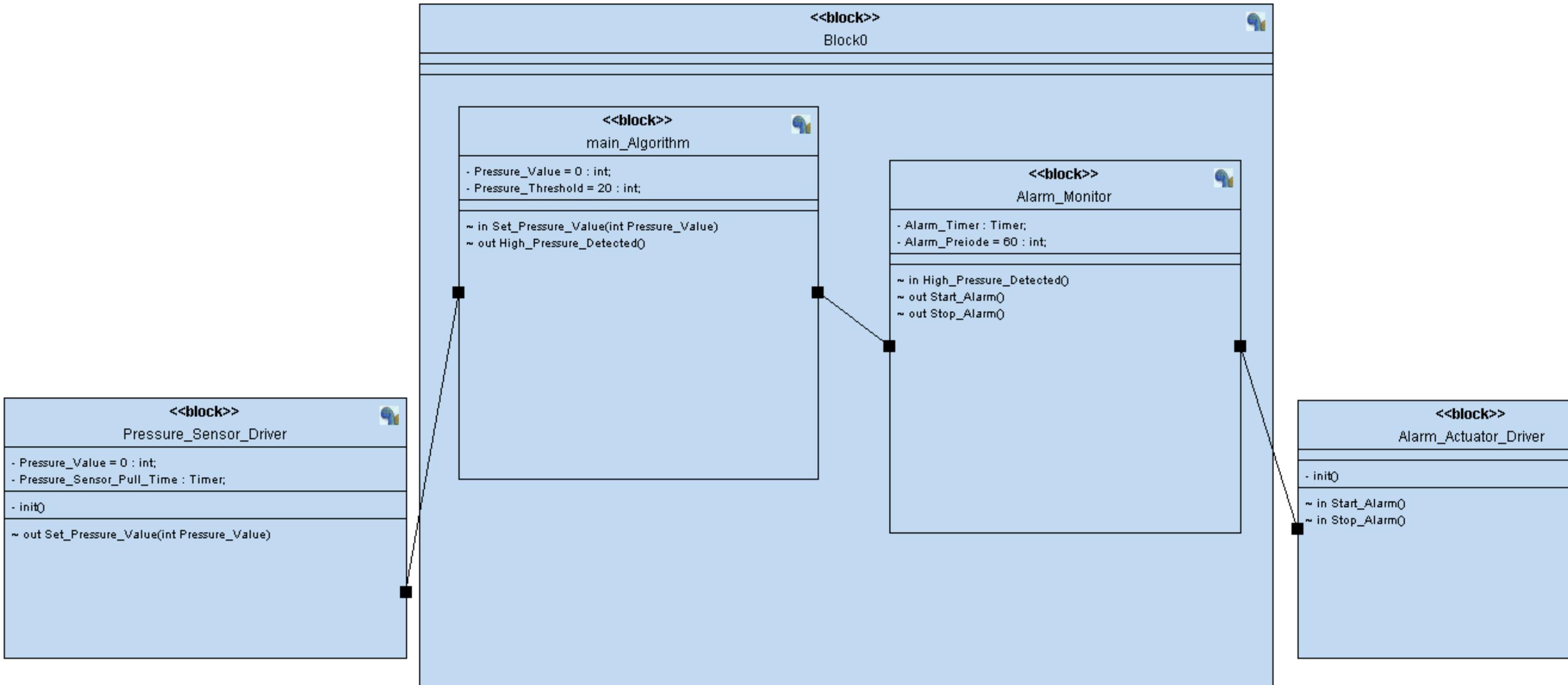
System Analysis

Sequence Diagram

3



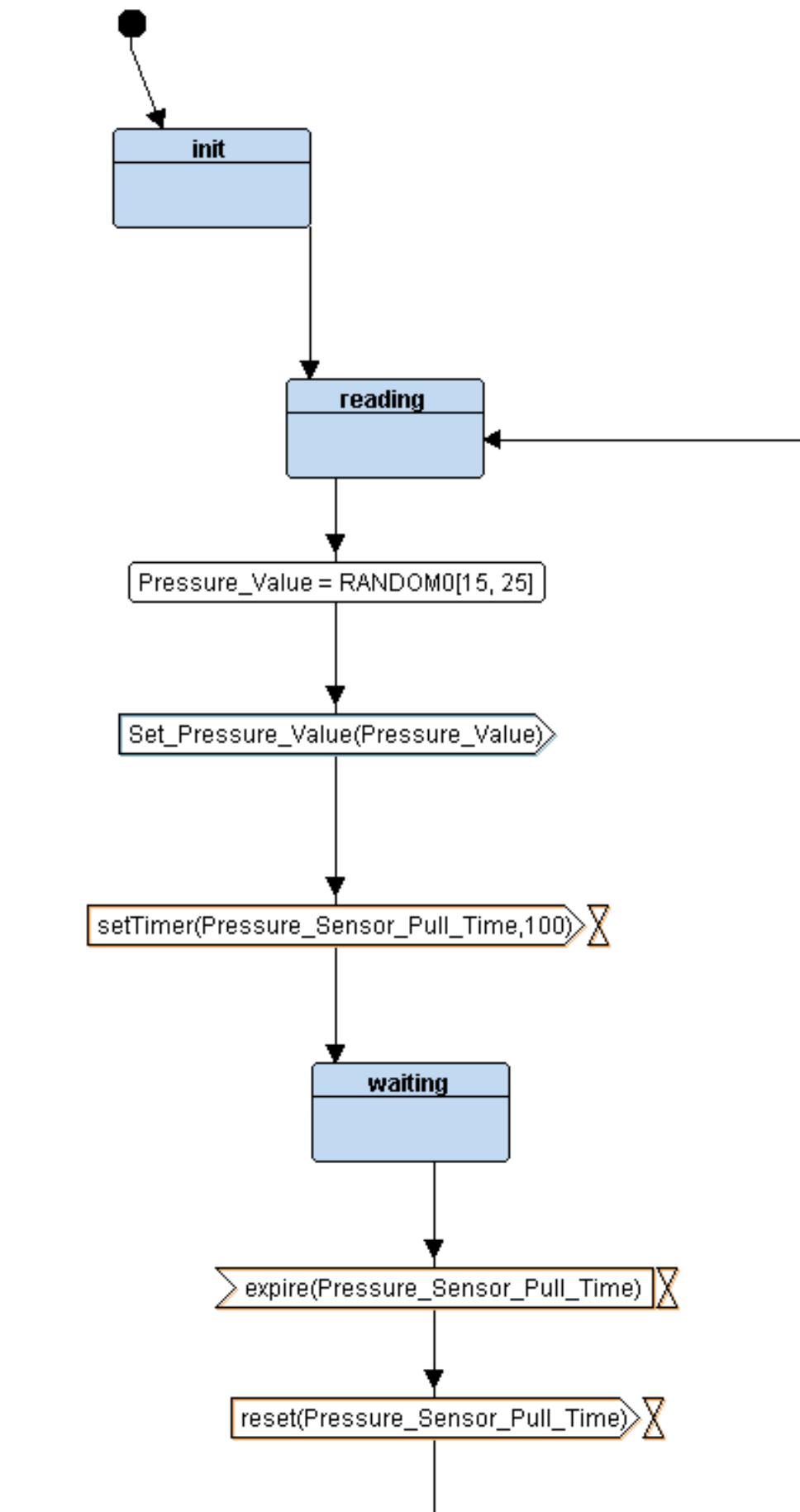
System Design



System Design



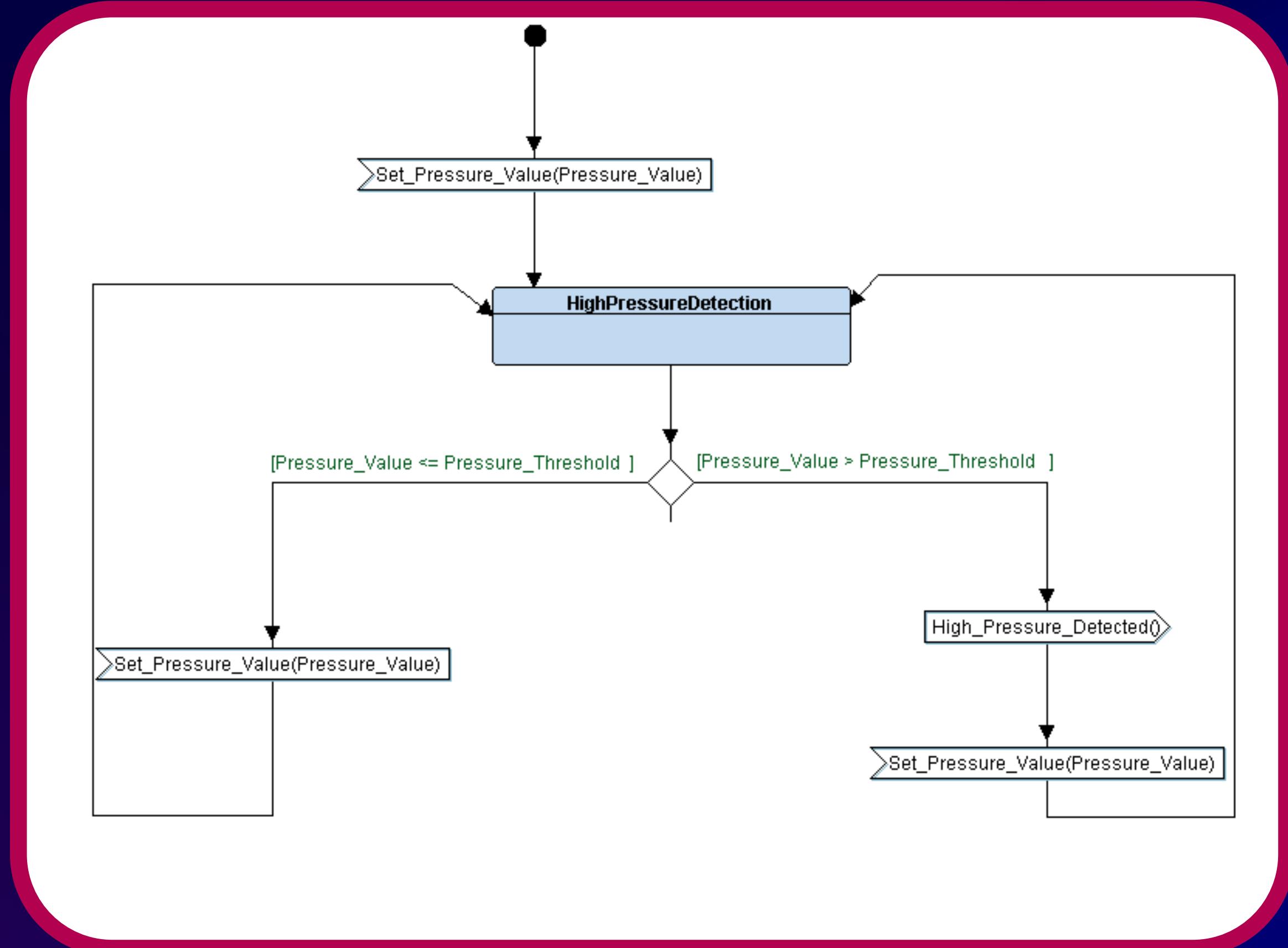
Pressure Sensor State Diagram



System Design

2

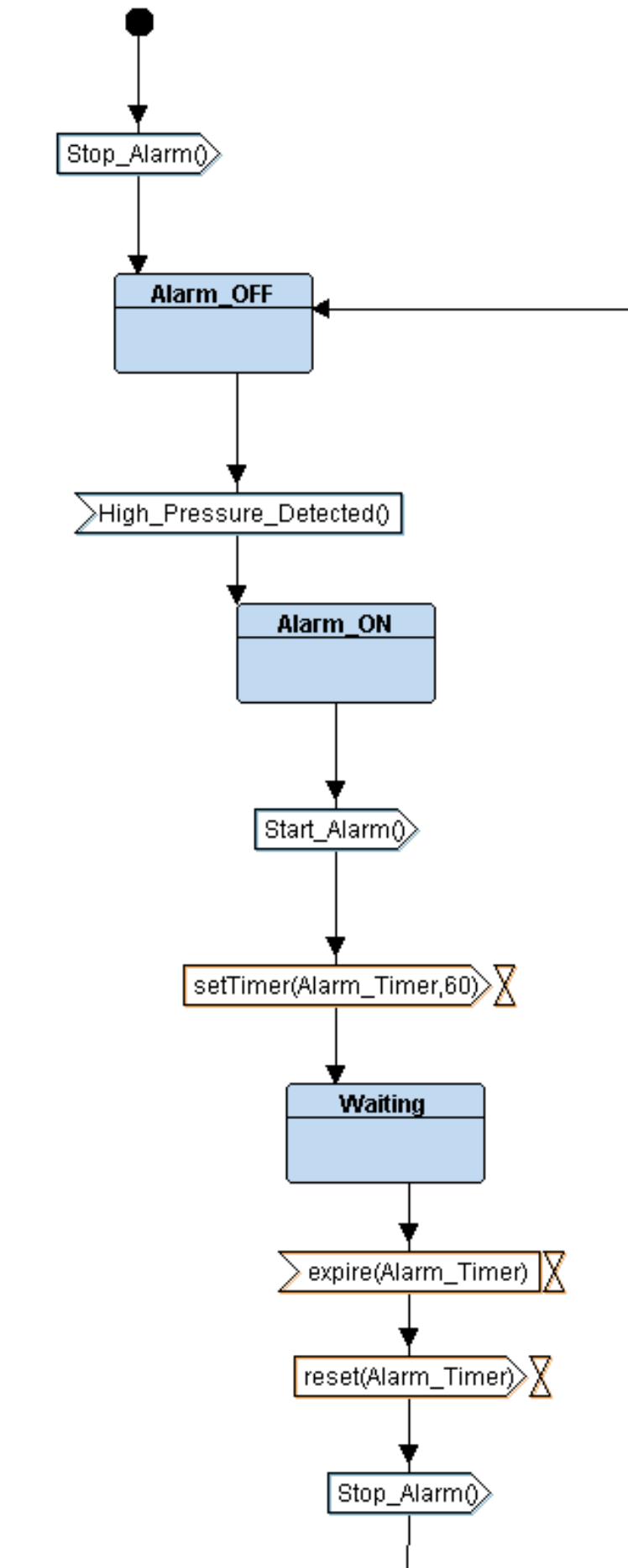
Main Program State Diagram



System Design

3

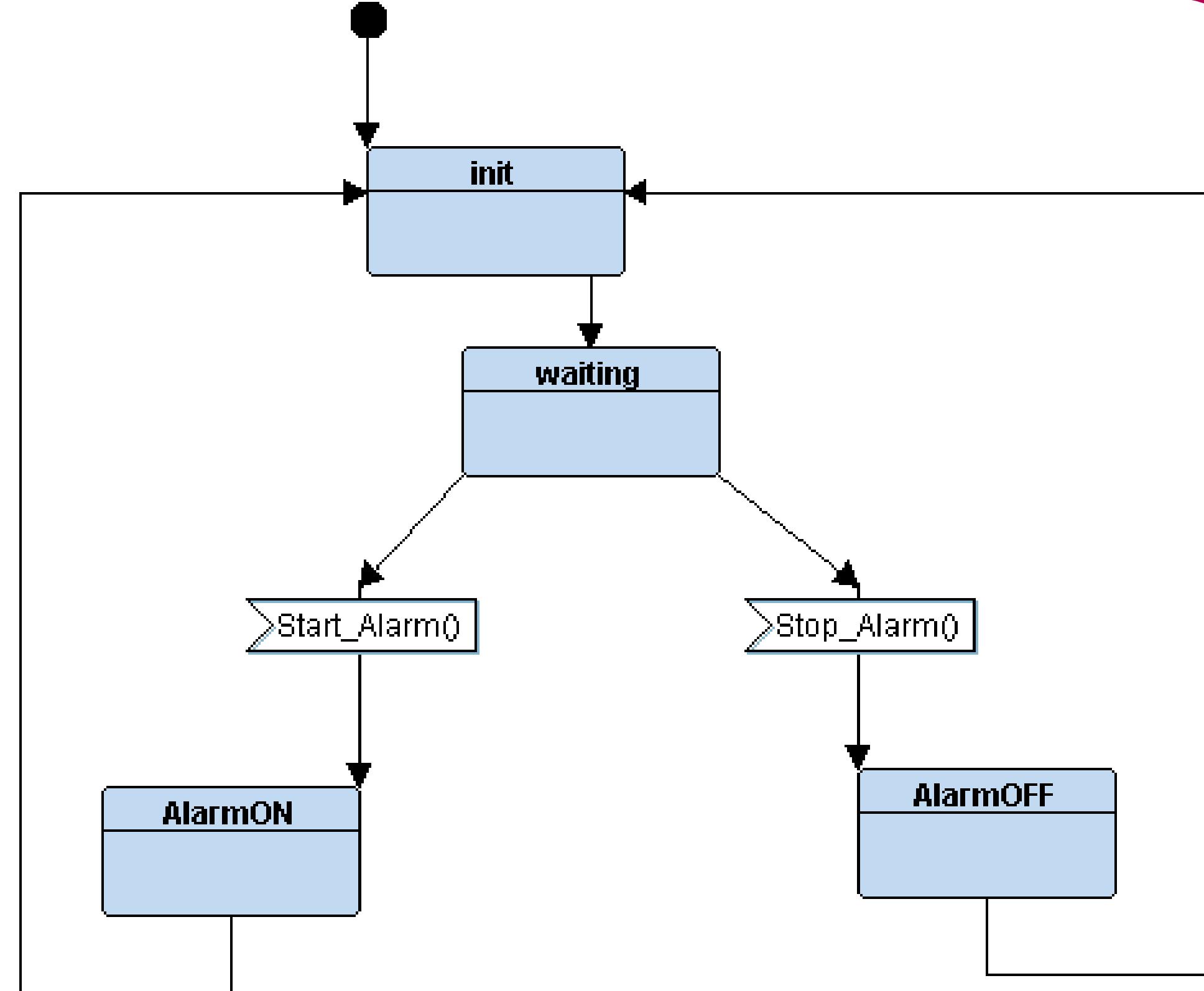
Alarm Monitor State Diagram



System Design

4

Alarm Actuator State Diagram



Design Simulation

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

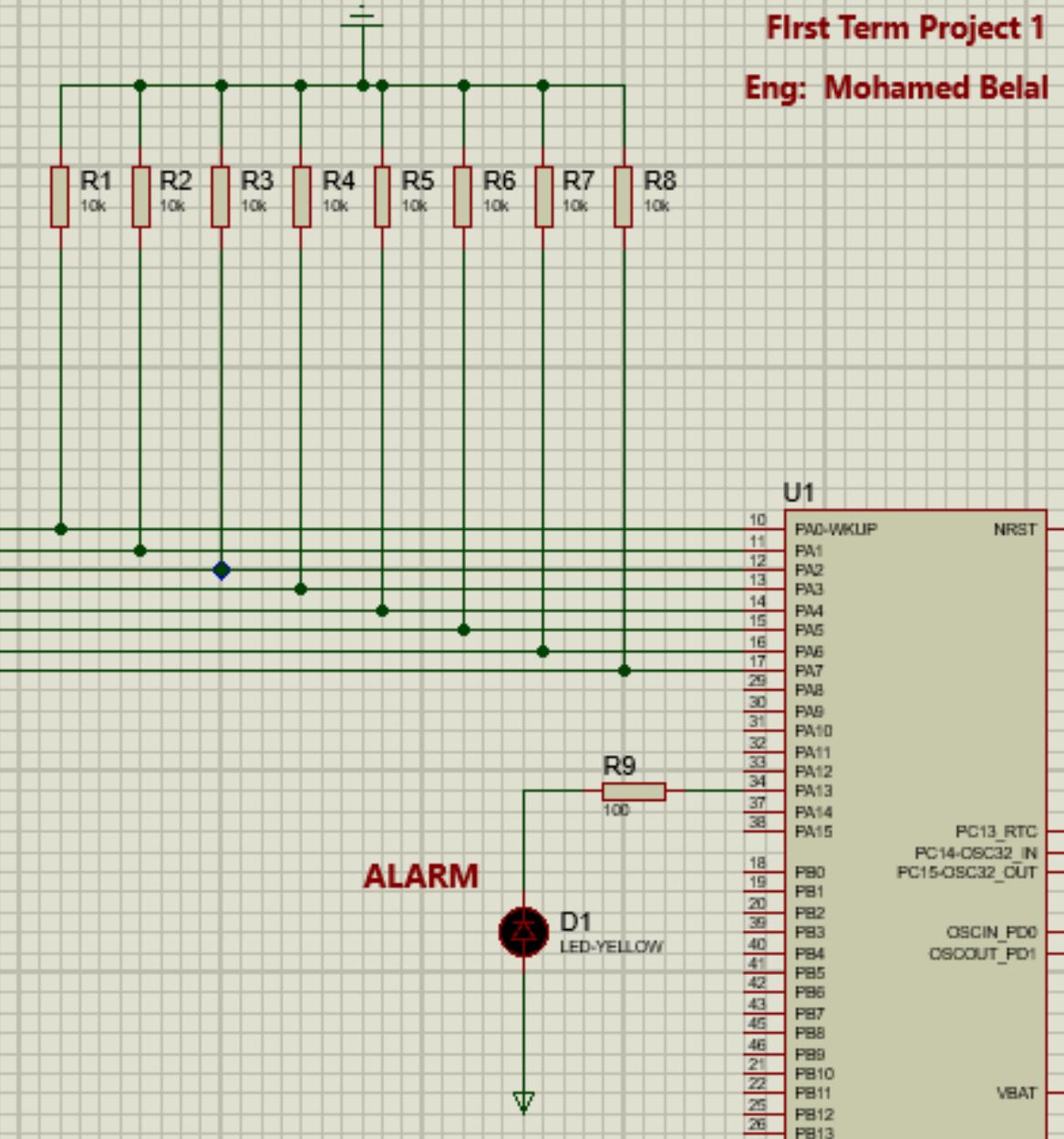
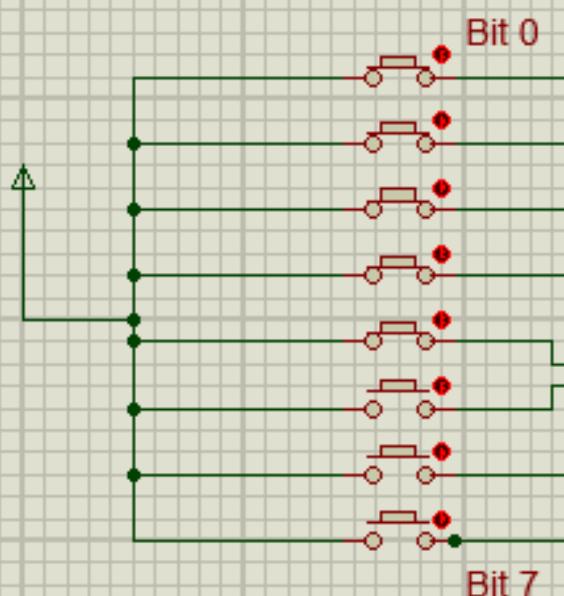
Mastering Embedded System Online Diploma (KS)

www.learn-in-depth.com

First Term Project 1

Eng: Mohamed Belal

Pressure Sensor



Source Code



main

```
1  /**
2  * ***** @project      : Pressure_Controller_MohamedBelal
3  * @file          : main.c
4  * @author        : mohamed belal
5  * @Created on    : Jul 19, 2023
6  * ****
7  */
8
9
10 #include <stdint.h>
11 #include <stdio.h>
12
13 #include "driver.h"
14 #include "Pressure_Sensor_Driver.h"
15 #include "main_Algorithm.h"
16 #include "Alarm_Monitor.h"
17 #include "Alarm_Actuator_Driver.h"
18
19 void setup()
20 {
21     /*
22     * 1 => init all the divers
23     * 2 => init iqr ....
24     * 3 => init hal us_driver dc_motor_driver
25     * 4 => init block
26     * 5 => set states pointers for each block
27     */
28     pressure_sensor_driver_init();
29     alarm_actuator_driver_init();
30
31     pressure_sensor_driver_state = state_calling(pressure_sensor_driver_reading);
32     main_algorithm_state = set_pressure_value;
33     alarm_monitor_state = state_calling(alarm_monitor_alarm_off);
34     alarm_actuator_driver_state = state_calling(alarm_actuator_driver_waiting);
35 }
36
37 int main()
38 {
39
40     GPIO_INITIALIZATION();
41     setup();
42     while (1)
43     {
44         Pressure_Sensor_Driver_state();
45         Main_Algorithm_state();
46         Alarm_Monitor_state();
47         Alarm_Actuator_Driver_state();
48     }
49     return 0;
50 }
```

Source Code

2

Pressure Sensor

```

1 /**
2 ****
3 * @project      : Pressure_Controller_MohamedBelal
4 * @file         : Pressure_Sensor_Driver.h
5 * @author       : mohamed belal
6 * @Created on   : Jul 19, 2023
7 ****
8 */
9
10 #include "Pressure_Sensor_Driver.h"
11
12 static int Pressure_Value = 0;
13 unsigned int Pressure_Sensor_Pull_Time = 100;
14 void (*Pressure_Sensor_Driver_state)();
15
16 void Pressure_Sensor_Driver_init()
17 {
18     // GPIO_INITIALIZATION();
19 }
20
21 STATE_Define(Pressure_Sensor_Driver_Reading)
22 {
23     // state_name
24     Pressure_Sensor_Driver_state_id = Pressure_Sensor_Driver_Reading;
25
26     Pressure_Value = getPressureVal();
27     Set_Pressure_Value(Pressure_Value);
28
29     Pressure_Sensor_Driver_state = STATE_calling(Pressure_Sensor_Driver_waiting);
30 }
31
32 STATE_Define(Pressure_Sensor_Driver_waiting)
33 {
34     // state_name
35     Pressure_Sensor_Driver_state_id = Pressure_Sensor_Driver_waiting;
36
37     Delay(60000);
38
39     Pressure_Sensor_Driver_state = STATE_calling(Pressure_Sensor_Driver_Reading);
40 }
41

```

```

1 /**
2 ****
3 * @project      : Pressure_Controller_MohamedBelal
4 * @file         : Pressure_Sensor_Driver.h
5 * @author       : mohamed belal
6 * @Created on   : Jul 19, 2023
7 ****
8 */
9
10 #ifndef PRESSURE_SENSOR_DRIVER_H_
11 #define PRESSURE_SENSOR_DRIVER_H_
12
13 #include "state.h"
14
15 // Define State
16 enum
17 {
18     Pressure_Sensor_Driver_Reading,
19     Pressure_Sensor_Driver_waiting,
20 }Pressure_Sensor_Driver_state_id;
21
22 // declare state function
23 void Pressure_Sensor_Driver_init();
24 STATE_Define(Pressure_Sensor_Driver_Reading);
25 STATE_Define(Pressure_Sensor_Driver_waiting);
26
27 // state pointer to function
28 extern void (*Pressure_Sensor_Driver_state)();
29
30 #endif /* PRESSURE_SENSOR_DRIVER_H_ */

```

Source Code

3

main
algorithm

```

1 /**
2 ****
3 * @project      : Pressure_Controller_MohamedBelal
4 * @file         : main_Algorithm.c
5 * @author       : mohamed belal
6 * @Created on   : Jul 19, 2023
7 ****
8 */
9 #include "main_Algorithm.h"
10 static int Main_Algorithm_Pressure_Value = 0;
11 static int Main_Algorithm_Pressure_Threshold = 20;
12 void (*Main_Algorithm_state)();
13
14 void Set_Pressure_Value(int Pressure_Val)
15 {
16     Main_Algorithm_Pressure_Value = Pressure_Val;
17     Main_Algorithm_state = STATE_calling(Main_Algorithm_High_Pressure_Detect);
18 }
19
20 // declare state function
21 STATE_Define(Main_Algorithm_High_Pressure_Detect)
22 {
23     // state_name
24     Main_Algorithm_state_id = Main_Algorithm_High_Pressure_Detect;
25
26
27     if (Main_Algorithm_Pressure_Value <= Main_Algorithm_Pressure_Threshold)
28     {
29         Main_Algorithm_state = STATE_calling(Main_Algorithm_High_Pressure_Detect);
30     }
31     else
32     {
33         High_Pressure_Detected();
34         Main_Algorithm_state = STATE_calling(Main_Algorithm_High_Pressure_Detect);
35     }
36 }
37

```

```

1 /**
2 ****
3 * @project      : Pressure_Controller_MohamedBelal
4 * @file         : main_Algorithm.h
5 * @author       : mohamed belal
6 * @Created on   : Jul 19, 2023
7 ****
8 */
9
10 #ifndef MAIN_ALGORITHM_H_
11 #define MAIN_ALGORITHM_H_
12 #include "state.h"
13
14 // Define State
15 enum
16 {
17     Main_Algorithm_High_Pressure_Detect,
18 } Main_Algorithm_state_id;
19
20
21 // declare state function
22 STATE_Define(Main_Algorithm_High_Pressure_Detect);
23
24
25 // state pointer to function
26 extern void (*Main_Algorithm_state)();
27
28 #endif /* MAIN_ALGORITHM_H_ */

```

Source Code

4

alarm
Monitor

```

1  /**
2  ****
3  * @project      : Pressure_Controller_MohamedBelal
4  * @file         : Alarm_Monitor.h
5  * @author       : mohamed belal
6  * @Created on   : Jul 19, 2023
7  ****
8 */
9
10 #ifndef ALARM_MONITOR_H_
11 #define ALARM_MONITOR_H_
12
13 #include "state.h"
14
15 // Define State
16 enum
17 {
18     Alarm_Monitor_alarm_off,
19     Alarm_Monitor_alarm_on,
20     Alarm_Monitor_waiting,
21 } Alarm_Monitor_state_id;
22
23
24
25 // declare state function
26 STATE_Define(Alarm_Monitor_alarm_off);
27 STATE_Define(Alarm_Monitor_alarm_on);
28 STATE_Define(Alarm_Monitor_waiting);
29
30
31
32 // state pointer to function
33 extern void (*Alarm_Monitor_state)();
34
35 #endif /* ALARM_MONITOR_H_ */

```

```

1  /**
2  ****
3  * @project      : Pressure_Controller_MohamedBelal
4  * @file         : Alarm_Monitor.c
5  * @author       : mohamed belal
6  * @Created on   : Jul 19, 2023
7  ****
8 */
9
10 #include "Alarm_Monitor.h"
11 unsigned int Alarm_Periode = 600000;
12
13 void (*Alarm_Monitor_state)();
14
15 void High_Pressure_Detected()
16 {
17     Alarm_Monitor_state = STATE_calling(Alarm_Monitor_alarm_on);
18 }
19
20 STATE_Define(Alarm_Monitor_alarm_off)
21 {
22     Alarm_Monitor_state_id = Alarm_Monitor_alarm_off;
23
24     Stop_Alarm();
25     Alarm_Monitor_state = STATE_calling(Alarm_Monitor_alarm_off);
26 }
27
28 STATE_Define(Alarm_Monitor_alarm_on)
29 {
30     Alarm_Monitor_state_id = Alarm_Monitor_alarm_on;
31
32     Start_Alarm();
33     Alarm_Monitor_state = STATE_calling(Alarm_Monitor_waiting);
34 }
35
36 STATE_Define(Alarm_Monitor_waiting)
37 {
38     Alarm_Monitor_state_id = Alarm_Monitor_waiting;
39     Stop_Alarm();
40     Delay(Alarm_Periode);
41
42     Alarm_Monitor_state = STATE_calling(Alarm_Monitor_alarm_off);
43 }
44

```

Source Code

alarm
Actuator

5

```

1  /**
2  ****
3  * @project      : Pressure_Controller_MohamedBelal
4  * @file         : Alarm_Actuator_Driver.h
5  * @author       : mohamed belal
6  * @Created on   : Jul 19, 2023
7  ****
8 */
9
10 #ifndef ALARM_ACTUATOR_DRIVER_H_
11 #define ALARM_ACTUATOR_DRIVER_H_
12
13 #include "state.h"
14
15 // Define State
16 enum
17 {
18     Alarm_Actuator_Driver_alarm_off,
19     Alarm_Actuator_Driver_alarm_on,
20     Alarm_Actuator_Driver_waiting,
21 } Alarm_Actuator_Driver_state_id;
22
23
24 // declare state function
25 void Alarm_Actuator_Driver_init();
26 STATE_Define(Alarm_Actuator_Driver_alarm_off);
27 STATE_Define(Alarm_Actuator_Driver_alarm_on);
28 STATE_Define(Alarm_Actuator_Driver_waiting);
29
30
31 // state pointer to function
32 extern void (*Alarm_Actuator_Driver_state)();
33
34 #endif /* ALARM_ACTUATOR_DRIVER_H_ */

```

```

1 /**
2 ****
3 * @project      : Pressure_Controller_MohamedBelal
4 * @file         : Alarm_Actuator_Driver.c
5 * @author       : mohamed belal
6 * @Created on   : Jul 19, 2023
7 ****
8 */
9
10 #include "Alarm_Actuator_Driver.h"
11 void (*Alarm_Actuator_Driver_state)();
12
13 void Stop_Alarm()
14 {
15
16     Alarm_Actuator_Driver_state = STATE_calling(Alarm_Actuator_Driver_alarm_off);
17 }
18
19 void Start_Alarm()
20 {
21
22     Alarm_Actuator_Driver_state = STATE_calling(Alarm_Actuator_Driver_alarm_on);
23 }
24
25 // declare state function
26 void Alarm_Actuator_Driver_init()
27 {
28
29 }
30
31 STATE_Define(Alarm_Actuator_Driver_waiting)
32 {
33     Alarm_Actuator_Driver_state_id = Alarm_Actuator_Driver_waiting;
34
35     Alarm_Actuator_Driver_state = STATE_calling(Alarm_Actuator_Driver_waiting);
36 }
37
38 STATE_Define(Alarm_Actuator_Driver_alarm_off)
39 {
40     Alarm_Actuator_Driver_state_id = Alarm_Actuator_Driver_alarm_off;
41
42     // we send (1) because of we connect Led with power
43     // then to reset the Led we send (1) as (5 volte) and another bolarty is (5 volte) this main that (0 volte) turn off Led
44     Set_Alarm_actuator(1);
45     Alarm_Actuator_Driver_state = STATE_calling(Alarm_Actuator_Driver_waiting);
46 }
47
48 STATE_Define(Alarm_Actuator_Driver_alarm_on)
49 {
50     Alarm_Actuator_Driver_state_id = Alarm_Actuator_Driver_alarm_on;
51
52     // we send (0) because of we connect Led with power
53     // then to set the led we send (0) as (0 volte) and another bolarty is (5 volte) this main that (5 volte) turn off Led
54     Set_Alarm_actuator(0);
55     Alarm_Actuator_Driver_state = STATE_calling(Alarm_Actuator_Driver_waiting);
56 }
57

```

Source Code

```
● ● ●  
1  /**  
2   ****  
3   * @project      : Pressure_Controller_MohamedBelal  
4   * @file         : state.h  
5   * @author       : mohamed belal  
6   * @Created on   : Jul 19, 2023  
7   ****  
8   */  
9  
10 #ifndef STATE_H_  
11 #define STATE_H_  
12  
13 #include <stdio.h>  
14 #include <stdlib.h>  
15 #include "driver.h"  
16  
17 #define STATE_Define(_stateFUN_) void ST_##_stateFUN_()
18 #define STATE_calling(_stateFUN_) ST_##_stateFUN_
19  
20 // state connection
21 //we use this for signal to connect eachblock with another
22 void Set_Pressure_Value(int Pressure_Val);
23 void High_Pressure_Detected();
24 void Stop_Alarm();
25 void Start_Alarm();
26  
27 #endif /* STATE_H_ */  
28
```

08

Source Code

7

alarm
Actuator

```

● ● ●
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

```

```

● ● ●
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 {
11     // Delay(40000);
12     // TOGGLE_BIT(GPIOA_ODR, 13);
13     // Delay(40000);
14     return (GPIOA_IDR & 0xFF);
15     // return 10;
16 }
17
18 void Set_Alarm_actuator(int i)
19 {
20     if (i == 1)
21     {
22         SET_BIT(GPIOA_ODR, 13);
23     }
24     else if (i == 0)
25     {
26         RESET_BIT(GPIOA_ODR, 13);
27     }
28 }
29
30 void GPIO_INITIALIZATION()
31 {
32     SET_BIT(APB2ENR, 2);
33     GPIOA_CRL &= 0xFF0FFFFF;
34     GPIOA_CRL |= 0x00000000;
35     GPIOA_CRH &= 0xFF0FFFFF;
36     GPIOA_CRH |= 0x22222222;
37 }
38

```

Sections for each object file



Main

```
PS D:\00_EMBEDDED SYSTEM\LEARN-IN-DEPTH\05_UNIT 5 FIRST TERM EXAM\PR

main.o: file format elf32-littlearm

Sections:
Idx Name      Size   VMA     LMA     File off  Align
0 .text       00000098 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
1 .data       00000000 00000000 00000000 000000cc 2**0
              CONTENTS, ALLOC, LOAD, DATA
2 .bss        00000000 00000000 00000000 000000cc 2**0
              ALLOC
3 .debug_info 00000018b 00000000 00000000 000000cc 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
4 .debug_abbrev 000000a5 00000000 00000000 00000257 2**0
              CONTENTS, READONLY, DEBUGGING
5 .debug_loc   00000058 00000000 00000000 000002fc 2**0
              CONTENTS, READONLY, DEBUGGING
6 .debug_aranges 00000020 00000000 00000000 00000354 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
7 .debug_line   000000b7 00000000 00000000 00000374 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
8 .debug_str    000002dc 00000000 00000000 0000042b 2**0
              CONTENTS, READONLY, DEBUGGING
9 .comment     00000012 00000000 00000000 00000707 2**0
              CONTENTS, READONLY
10 .ARM.attributes 00000033 00000000 00000000 00000719 2**0
              CONTENTS, READONLY
11 .debug_frame 00000048 00000000 00000000 0000074c 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
```



Pressure Sensor Driver

```
PS D:\00_EMBEDDED SYSTEM\LEARN-IN-DEPTH\05_UNIT 5 FIRST TERM EXAM\PR

Pressure_Sensor_Driver.o: file format elf32-littlearm

Sections:
Idx Name      Size   VMA     LMA     File off  Align
0 .text       00000084 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
1 .data       00000004 00000000 00000000 000000b8 2**2
              CONTENTS, ALLOC, LOAD, DATA
2 .bss        00000004 00000000 00000000 000000bc 2**2
              ALLOC
3 .debug_info 00000122 00000000 00000000 000000bc 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
4 .debug_abbrev 000000a1 00000000 00000000 000001de 2**0
              CONTENTS, READONLY, DEBUGGING
5 .debug_loc   00000084 00000000 00000000 0000027f 2**0
              CONTENTS, READONLY, DEBUGGING
6 .debug_aranges 00000020 00000000 00000000 00000303 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
7 .debug_line   00000070 00000000 00000000 00000323 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
8 .debug_str    00000213 00000000 00000000 00000393 2**0
              CONTENTS, READONLY, DEBUGGING
9 .comment     00000012 00000000 00000000 000005a6 2**0
              CONTENTS, READONLY
10 .ARM.attributes 00000033 00000000 00000000 000005b8 2**0
              CONTENTS, READONLY
11 .debug_frame 00000060 00000000 00000000 000005ec 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
```



main Algorithm

```
PS D:\00_EMBEDDED SYSTEM\LEARN-IN-DEPTH\05_UNIT 5 FIRST TERM EXAM\PR

main_Algorithm.o: file format elf32-littlearm

Sections:
Idx Name      Size   VMA     LMA     File off  Align
0 .text       00000088 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
1 .data       00000004 00000000 00000000 000000bc 2**2
              CONTENTS, ALLOC, LOAD, DATA
2 .bss        00000004 00000000 00000000 000000c0 2**2
              ALLOC
3 .debug_info 0000011a 00000000 00000000 000000c0 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
4 .debug_abbrev 000000b4 00000000 00000000 000001da 2**0
              CONTENTS, READONLY, DEBUGGING
5 .debug_loc   00000064 00000000 00000000 0000028e 2**0
              CONTENTS, READONLY, DEBUGGING
6 .debug_aranges 00000020 00000000 00000000 000002f2 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
7 .debug_line   0000005e 00000000 00000000 00000312 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
8 .debug_str    000001df 00000000 00000000 00000370 2**0
              CONTENTS, READONLY, DEBUGGING
9 .comment     00000012 00000000 00000000 0000054f 2**0
              CONTENTS, READONLY
10 .ARM.attributes 00000033 00000000 00000000 00000561 2**0
              CONTENTS, READONLY
11 .debug_frame 00000048 00000000 00000000 00000594 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
```



Alarm Monitor

```
PS D:\00_EMBEDDED SYSTEM\LEARN-IN-DEPTH\05_UNIT 5 FIRST TERM EXAM\PR

Alarm_Monitor.o: file format elf32-littlearm

Sections:
Idx Name      Size   VMA     LMA     File off  Align
0 .text       000000b0 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
1 .data       00000004 00000000 00000000 000000e4 2**2
              CONTENTS, ALLOC, LOAD, DATA
2 .bss        00000000 00000000 00000000 000000e8 2**0
              ALLOC
3 .debug_info 0000012c 00000000 00000000 000000e8 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
4 .debug_abbrev 00000092 00000000 00000000 00000214 2**0
              CONTENTS, READONLY, DEBUGGING
5 .debug_loc   000000b0 00000000 00000000 000002a6 2**0
              CONTENTS, READONLY, DEBUGGING
6 .debug_aranges 00000020 00000000 00000000 00000356 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
7 .debug_line   00000064 00000000 00000000 00000376 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
8 .debug_str    000001e9 00000000 00000000 000003da 2**0
              CONTENTS, READONLY, DEBUGGING
9 .comment     00000012 00000000 00000000 000005c3 2**0
              CONTENTS, READONLY
10 .ARM.attributes 00000033 00000000 00000000 000005d5 2**0
              CONTENTS, READONLY
11 .debug_frame 0000007c 00000000 00000000 00000608 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
```



Alarm Actuator Driver

```
PS D:\00_EMBEDDED SYSTEM\LEARN-IN-DEPTH\05_UNIT 5 FIRST TERM EXAM\PR

Alarm_Actuator_Driver.o: file format elf32-littlearm

Sections:
Idx Name      Size   VMA     LMA     File off  Align
0 .text       000000d0 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
1 .data       00000000 00000000 00000000 00000104 2**0
              CONTENTS, ALLOC, LOAD, DATA
2 .bss        00000000 00000000 00000000 00000104 2**0
              ALLOC
3 .debug_info 00000144 00000000 00000000 00000104 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
4 .debug_abbrev 00000092 00000000 00000000 00000248 2**0
              CONTENTS, READONLY, DEBUGGING
5 .debug_loc   000000b8 00000000 00000000 000002da 2**0
              CONTENTS, READONLY, DEBUGGING
6 .debug_aranges 00000020 00000000 00000000 000003e2 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
7 .debug_line   00000077 00000000 00000000 00000402 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
8 .debug_str    0000023e 00000000 00000000 00000479 2**0
              CONTENTS, READONLY, DEBUGGING
9 .comment     00000012 00000000 00000000 000006b7 2**0
              CONTENTS, READONLY
10 .ARM.attributes 00000033 00000000 00000000 000006c9 2**0
              CONTENTS, READONLY
11 .debug_frame 000000a8 00000000 00000000 000006fc 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
```



startup

```
PS D:\00_EMBEDDED SYSTEM\LEARN-IN-DEPTH\05_UNIT 5 FIRST TERM EXAM\PR

startup.o: file format elf32-littlearm

Sections:
Idx Name      Size   VMA     LMA     File off  Align
0 .text       000000c0 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
1 .data       00000000 00000000 00000000 000000f4 2**0
              CONTENTS, ALLOC, LOAD, DATA
2 .bss        00000000 00000000 00000000 000000f4 2**0
              ALLOC
3 .vectors   0000001c 00000000 00000000 000000f4 2**2
              CONTENTS, ALLOC, LOAD, RELOC, DATA
4 .debug_info 00000164 00000000 00000000 00000110 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
5 .debug_abbrev 000000e1 00000000 00000000 00000274 2**0
              CONTENTS, READONLY, DEBUGGING
6 .debug_loc   00000064 00000000 00000000 00000355 2**0
              CONTENTS, READONLY, DEBUGGING
7 .debug_aranges 00000020 00000000 00000000 000003b9 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
8 .debug_line   00000069 00000000 00000000 000003d9 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
9 .debug_str    00000107 00000000 00000000 00000442 2**0
              CONTENTS, READONLY, DEBUGGING
10 .comment    00000012 00000000 00000000 00000549 2**0
              CONTENTS, READONLY
11 .ARM.attributes 00000033 00000000 00000000 0000055b 2**0
              CONTENTS, READONLY
12 .debug_frame 0000004c 00000000 00000000 00000590 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
```

Symbol Table for each object file



Main

```
PS D:\00_EMBEDDED_System\learn-in-depth\05_unit 5
    U Alarm_Actuator_Driver_init
    U Alarm_Actuator_Driver_state
00000001 C Alarm_Actuator_Driver_state_id
    U Alarm_Monitor_state
00000001 C Alarm_Monitor_state_id
    U GPIO_INITIALIZATION
00000058 T main
    U Main_Algorithm_state
00000001 C Main_Algorithm_state_id
    U Pressure_Sensor_Driver_init
    U Pressure_Sensor_Driver_state
00000001 C Pressure_Sensor_Driver_state_id
    U Set_Pressure_Value
00000000 T setup
    U ST_Alarm_Actuator_Driver_waiting
    U ST_Alarm_Monitor_alarm_off
    U ST_Pressure_Sensor_Driver_Reading
```



Pressure Sensor Driver

```
PS D:\00_EMBEDDED_System\learn-in-depth\05_unit 5 f
    U Delay
    U getPressureVal
00000000 T Pressure_Sensor_Driver_init
00000004 C Pressure_Sensor_Driver_state
00000001 C Pressure_Sensor_Driver_state_id
00000000 D Pressure_Sensor_Pull_Time
00000000 b Pressure_Value
    U Set_Pressure_Value
0000000c T ST_Pressure_Sensor_Driver_Reading
00000054 T ST_Pressure_Sensor_Driver_waiting
```



main Algorithm

```
PS D:\00_EMBEDDED_System\learn-in-depth\05_unit 5
    U High_Pressure_Detected
00000000 d Main_Algorithm_Pressure_Threshold
00000000 b Main_Algorithm_Pressure_Value
00000004 C Main_Algorithm_state
00000001 C Main_Algorithm_state_id
00000000 T Set_Pressure_Value
00000030 T ST_Main_Algorithm_High_Pressure_Detect
```



Alarm Monitor

```
PS D:\00_EMBEDDED_System\learn-in-depth\
00000004 C Alarm_Monitor_state
00000001 C Alarm_Monitor_state_id
00000000 D Alarm_Periode
    U Delay
00000000 T High_Pressure_Detected
0000001c T ST_Alarm_Monitor_alarm_off
00000048 T ST_Alarm_Monitor_alarm_on
00000074 T ST_Alarm_Monitor_waiting
    U Start_Alarm
    U Stop_Alarm
```



Alarm Actuator Driver

```
PS D:\00_EMBEDDED_System\learn-in-depth\05_unit 5 first term exam\p
    T Alarm_Actuator_Driver_init
00000004 C Alarm_Actuator_Driver_state
00000001 C Alarm_Actuator_Driver_state_id
    U Set_Alarm_actuator
00000070 T ST_Alarm_Actuator_Driver_alarm_off
000000a0 T ST_Alarm_Actuator_Driver_alarm_on
00000044 T ST_Alarm_Actuator_Driver_waiting
0000001c T Start_Alarm
00000000 T Stop_Alarm
```



startup

```
PS D:\00_EMBEDDED_System\learn-in-depth\05_unit 5 first term exam\p
startup.o:   file format elf32-littlearm

Sections:
Idx Name          Size    VMA     LMA     File off  Align
 0 .text         000000c0  00000000  00000000  00000034  2**2
                CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data         00000000  00000000  00000000  000000f4  2**0
                CONTENTS, ALLOC, LOAD, DATA
 2 .bss          00000000  00000000  00000000  000000f4  2**0
                ALLOC
 3 .vectors      0000001c  00000000  00000000  000000f4  2**2
                CONTENTS, ALLOC, LOAD, RELOC, DATA
 4 .debug_info   00000164  00000000  00000000  00000110  2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
 5 .debug_abbrev 000000e1  00000000  00000000  00000274  2**0
                CONTENTS, READONLY, DEBUGGING
 6 .debug_loc    00000064  00000000  00000000  00000355  2**0
                CONTENTS, READONLY, DEBUGGING
 7 .debug_aranges 00000020  00000000  00000000  000003b9  2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_line   00000069  00000000  00000000  000003d9  2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
 9 .debug_str    00000107  00000000  00000000  00000442  2**0
                CONTENTS, READONLY, DEBUGGING
10 .comment     00000012  00000000  00000000  00000549  2**0
                CONTENTS, READONLY
11 .ARM.attributes 00000033  00000000  00000000  0000055b  2**0
                CONTENTS, READONLY
12 .debug_frame  0000004c  00000000  00000000  00000590  2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
```

sections

```
Windows PowerShell
PS D:\00_EMBEDDED SYSTEM LEARN-IN-DEPTH\05_UNIT 5 FIRST TERM EXAM\PROJECT_1_PRESSURE_CONTROLLER\CODE> arm-none-eabi-objdump.exe -h pressure_controller_cortex_m3.elf ^

pressure_controller_cortex_m3.elf:      file format elf32-littlearm

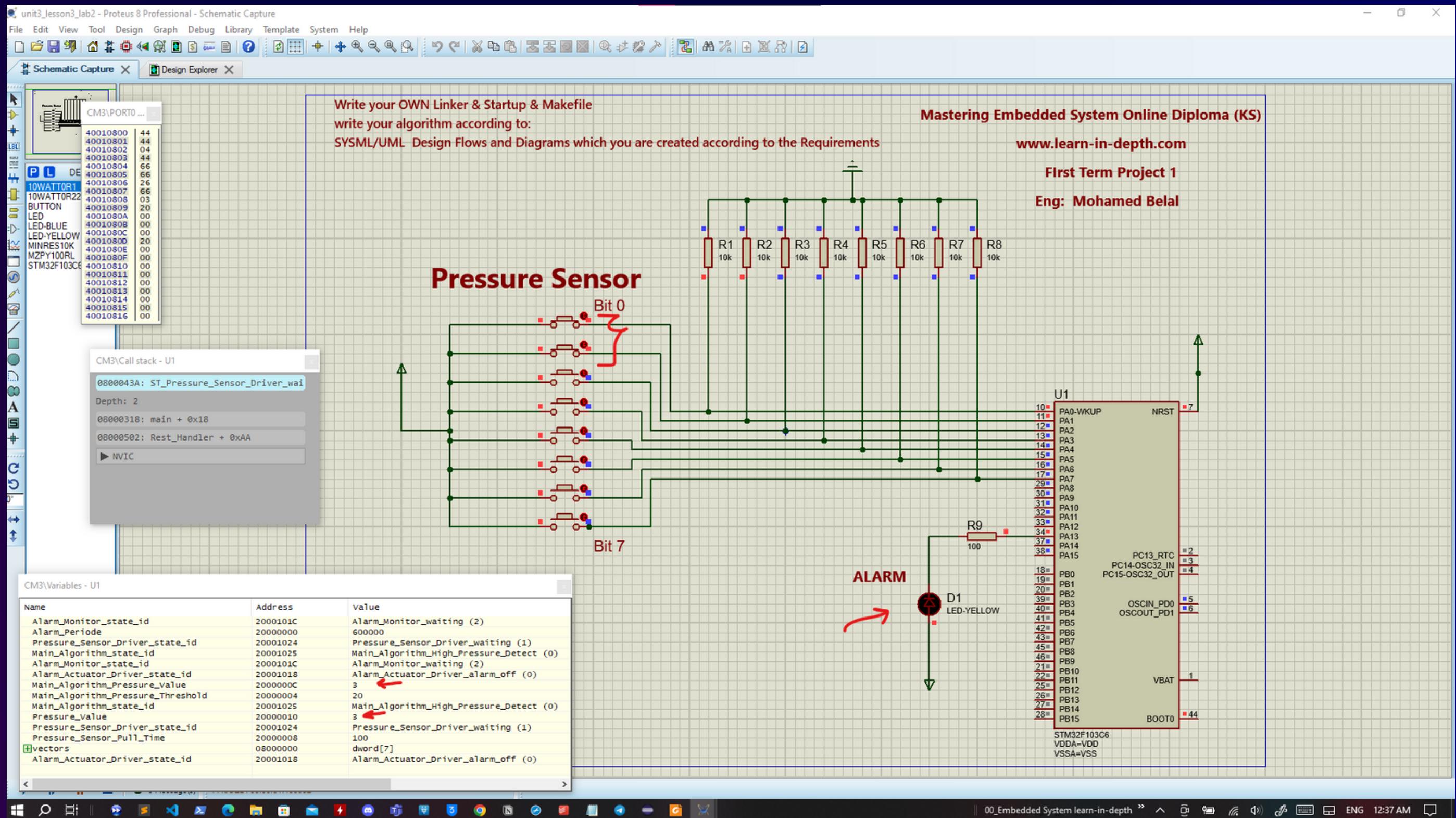
Sections:
Idx Name      Size    VMA     LMA     File off  Align
 0 .text      0000050c 08000000 08000000 00008000 2**2
              CONTENTS, ALLOC, LOAD, READONLY, CODE
 1 .data      0000000c 20000000 0800050c 00010000 2**2
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss      00001024 2000000c 08000518 0001000c 2**2
              ALLOC
 3 .debug_info 0000089e 00000000 00000000 0001000c 2**0
              CONTENTS, READONLY, DEBUGGING
 4 .debug_abbrev 0000049c 00000000 00000000 000108aa 2**0
              CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   00000424 00000000 00000000 00010d46 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 000000e0 00000000 00000000 0001116a 2**0
              CONTENTS, READONLY, DEBUGGING
 7 .debug_line   00000362 00000000 00000000 0001124a 2**0
              CONTENTS, READONLY, DEBUGGING
 8 .debug_str    000004d1 00000000 00000000 000115ac 2**0
              CONTENTS, READONLY, DEBUGGING
 9 .comment     00000011 00000000 00000000 00011a7d 2**0
              CONTENTS, READONLY
10 .ARM.attributes 00000033 00000000 00000000 00011a8e 2**0
              CONTENTS, READONLY
11 .debug_frame 000002d8 00000000 00000000 00011ac4 2**2
              CONTENTS, READONLY, DEBUGGING
```

Symbol Table

```
Windows PowerShell
PS D:\00_EMBEDDED SYSTEM LEARN-IN-DEPTH\05_UNIT 5 FIRST TERM EXAM\PROJECT_1_PRESSURE_CONTROLLER\CODE> arm-none-eabi-nm.exe pressure_controller_cortex_m3.elf
20000014 B _E_bss
2000000c D _E_data
0800050c T _E_text
2000000c B _S_bss
20000000 D _S_data
20001014 B _stack_top
08000054 T Alarm_Actuator_Driver_init
20001014 B Alarm_Actuator_Driver_state
20001018 B Alarm_Actuator_Driver_state_id
20001020 B Alarm_Monitor_state
2000101c B Alarm_Monitor_state_id
20000000 D Alarm_Periode
0800044c W Bus_Handler
0800044c T Default_Handler
0800019c T Delay
080001c0 T getPressureVal
08000228 T GPIO_INITIALIZATION
0800044c W H_Fault_Handler
080000ec T High_Pressure_Detected
08000300 T main
20000004 d Main_Algorithm_Pressure_Threshold
2000000c b Main_Algorithm_Pressure_Value
20001028 B Main_Algorithm_state
20001025 B Main_Algorithm_state_id
0800044c W MM_Handler
0800044c W NMI_Handler
080003c8 T Pressure_Sensor_Driver_init
2000102c B Pressure_Sensor_Driver_state
20001024 B Pressure_Sensor_Driver_state_id
20000008 D Pressure_Sensor_Pull_Time
20000010 b Pressure_Value
08000458 T Rest_Handler
080001d8 T Set_Alarm_actuator
08000340 T Set_Pressure_Value
080002a8 T setup
0800008c T ST_Alarm_Actuator_Driver_alarm_off
080000bc T ST_Alarm_Actuator_Driver_alarm_on
08000060 T ST_Alarm_Actuator_Driver_waiting
08000108 T ST_Alarm_Monitor_alarm_off
08000134 T ST_Alarm_Monitor_alarm_on
08000160 T ST_Alarm_Monitor_waiting
08000370 T ST_Main_Algorithm_High_Pressure_Detect
080003d4 T ST_Pressure_Sensor_Driver_Reading
0800041c T ST_Pressure_Sensor_Driver_waiting
08000038 T Start_Alarm
0800001c T Stop_Alarm
0800044c W Usage_Handler
08000000 T vectors
```

Hardware Simulation

1
If
pressureVal <= Threshold
Alarm OFF
Threshold = 20



Hardware Simulation

2
If
pressureVal > Threshold
Alarm ON
Threshold = 20

