

[Mastering Embedded System Online Diploma](http://www.learn-in-depth.com)

[www.learn-in-depth.com](http://www.learn-in-depth.com)

**First Term (Final Project 1)**

**High Pressure Detection**

**Eng. Salma Sherif**

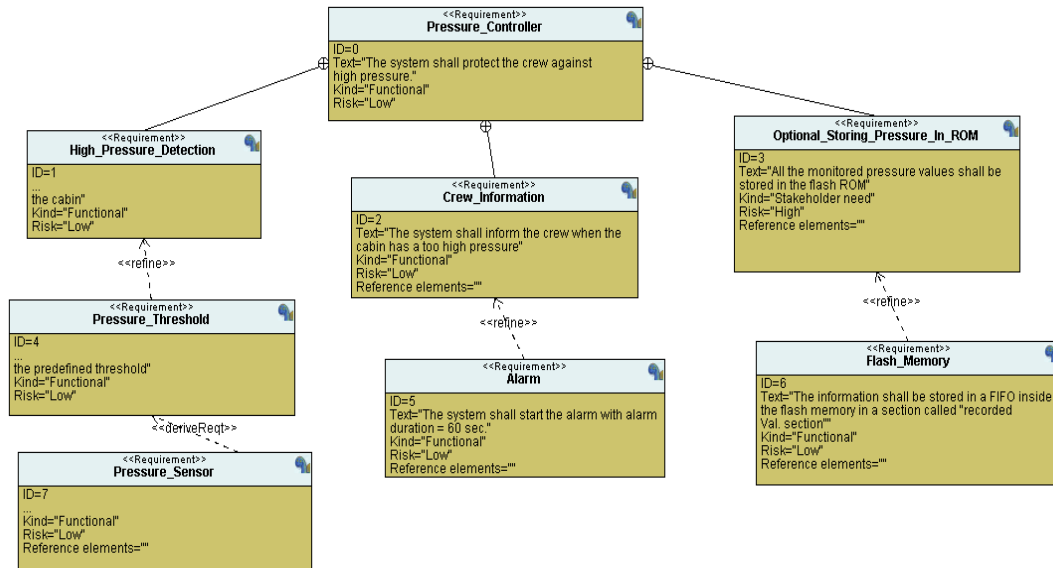
## **Overview:**

**This is a report about the Pressure Controller System in an airplane which informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin. The alarm duration equals 60 seconds. This project is done from scratch, with a written linker script, a startup and a make file.**

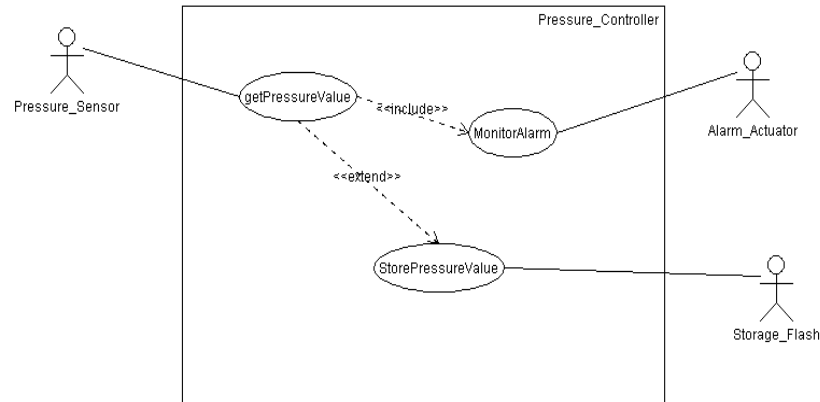
**The report will include UML diagrams, screenshot from the simulation demo and the code.**

**STM32F106C6 microcontroller is used in the implementation.**

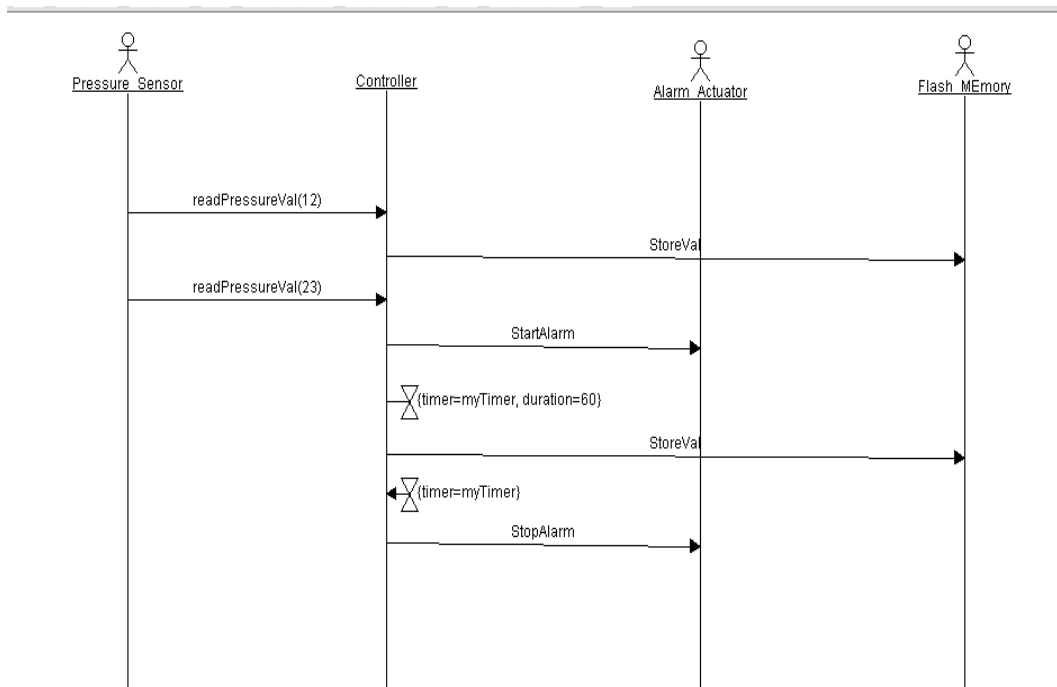
## 1. Requirement Diagram



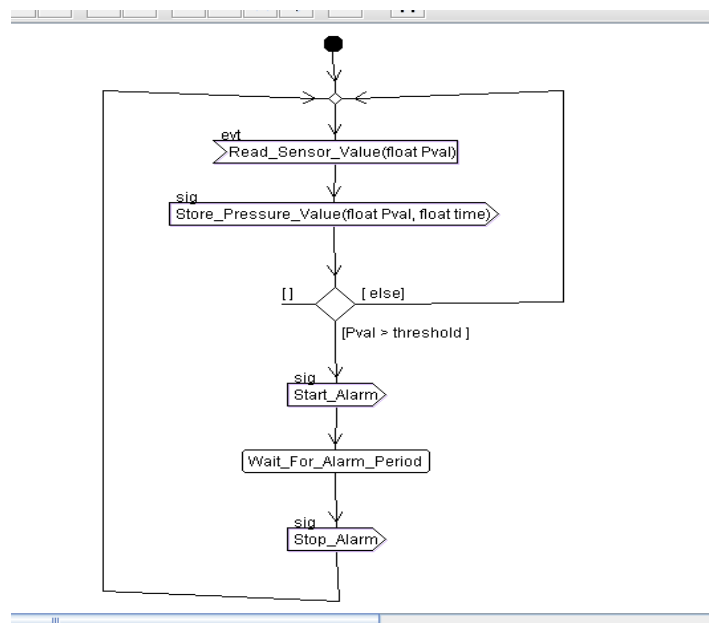
## 2. Use Case Diagram



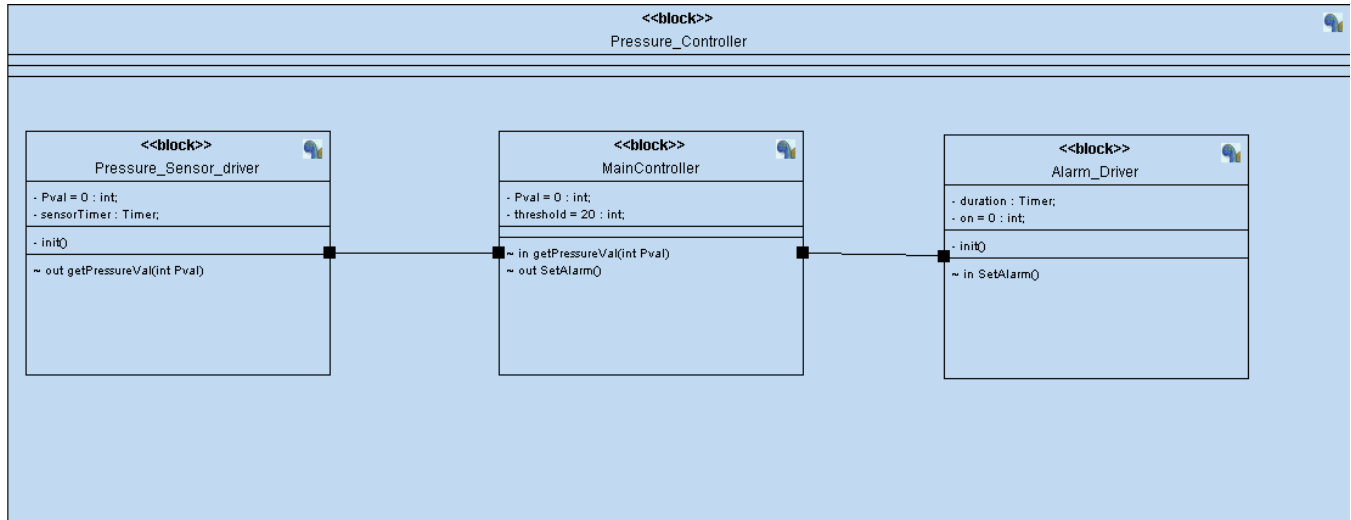
### 3. Sequence Diagram



### 4. Activity Diagram

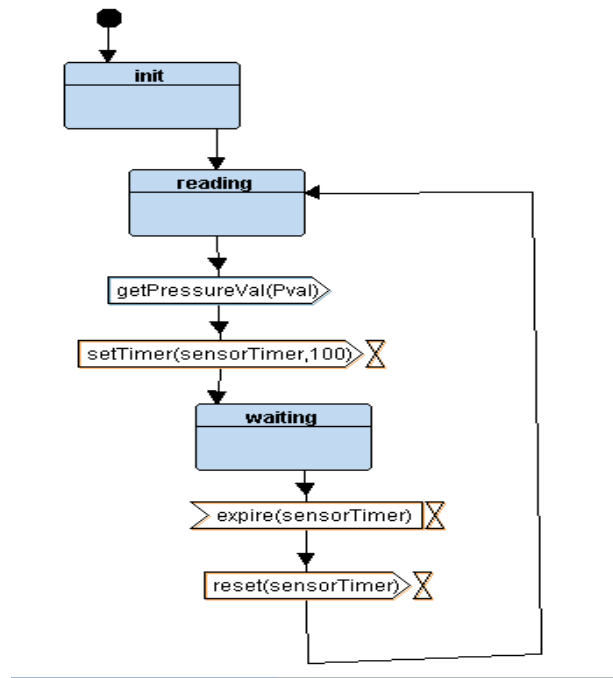


## 5. System Design



## 6. State Machines with its corresponding .c & .h files

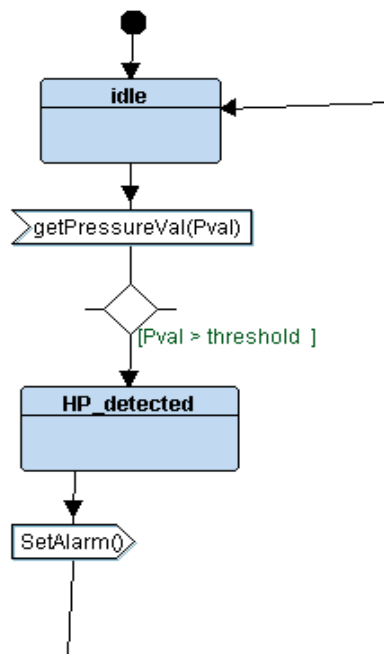
### 6.1. Pressure Sensor Driver



```
1 > /* ...
2
3 #include "PressureSensorDriver.h"
4 int getPressureVal();
5 //int generate_random(int l, int r, int c);
6
7 int pVal = 0;
8 void (*PS_state)();
9
10 void PS_init(){
11     PS_state = STATE(PressureSensorDriver_reading);
12     //printf("INIT PS \n");
13 }
14
15 int getPressureVal(){
16     //return generate_random(15,25,1);
17     return (GPIOA_IDR & 0xFF);
18 }
19
20 STATE_define(PressureSensorDriver_waiting){
21     //printf("Sensor state : Waiting.....\n");
22     Delay(100000);
23     PS_state = STATE(PressureSensorDriver_reading);
24 }
25
26 STATE_define(PressureSensorDriver_reading){
27     PS_states_id = PressureSensorDriver_reading;
28     // printf("Sensor state : reading.....\n");
29     pVal = getPressureVal();
30     get_pressure_value(pVal);
31     PS_state = STATE(PressureSensorDriver_waiting);
32 }
33
34 }
```

```
1 /*
2  * PressureSensor.h
3  *
4  * Created on: Sep 5, 2021
5  * Author: Salma Sherif
6  */
7
8 #ifndef PRESSURESENSORDRIVER_H_
9 #define PRESSURESENSORDRIVER_H_
10
11 #include "states.h"
12 #include "driver.h"
13 // #include "stdlib.h"
14
15 enum{
16     PS_reading,
17     PS_waiting
18 }PS_states_id;
19
20 void PS_init();
21 STATE_define(PressureSensorDriver_reading);
22
23 extern void (*PS_state)();
24 #endif /* PRESSURESENSORDRIVER_H_ */
25
26
```

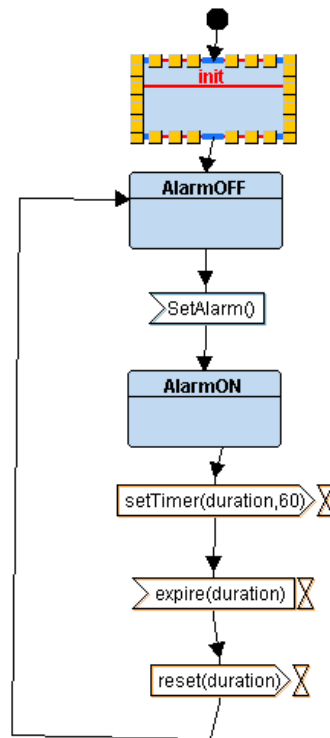
## 6.2. Main Controller



```
C MainController.c x C AlarmDriver.c C PressureSensorDriver.c C
D: > embedded diploma > REPO > embedded-systems-diploma > First_Term_Project1
1 > /* ...
7 < #include "MainController.h"
8
9 int threshold = 20;
10
11 void (*C_state)() = STATE(idle);
12
13
14 void get_pressure_value(int P){
15     // printf("Sensor reads : %d\n", P);
16     if (P > 20){
17         C_state = STATE(HP_detected);
18     }else{
19         C_state = STATE(idle);
20     }
21 }
22
23
24 STATE_define(HP_detected){
25     C_id = HP_detected;
26     // printf("Controller in HP detected state .....n");
27     Set_Alarm_actuator(1);
28     C_state = STATE(idle);
29 }
30
31
32 STATE_define(idle){
33     Set_Alarm_actuator(0);
34     C_id = idle;
35     // printf("Controller in idle state .....n");
36 }
37
38 }
```

```
D: > embedded diploma > REPO > embedded-systems
1 /*
2  * MainController.h
3  *
4  * Created on: Sep 5, 2021
5  * Author: Salma Sherif
6  */
7
8 #ifndef MAINCONTROLLER_H_
9 #define MAINCONTROLLER_H_
10
11 #include "driver.h"
12 #include "states.h"
13 #include "PressureSensorDriver.h"
14
15
16 enum{
17     idle,
18     HP_detected
19 }C_id;
20
21 STATE_define(HP_detected);
22 STATE_define(idle);
23
24
25
26 extern void (*C_state)();
27
28 #endif /* MAINCONTROLLER_H_ */
29
```

### 6.3. Alarm Driver



```

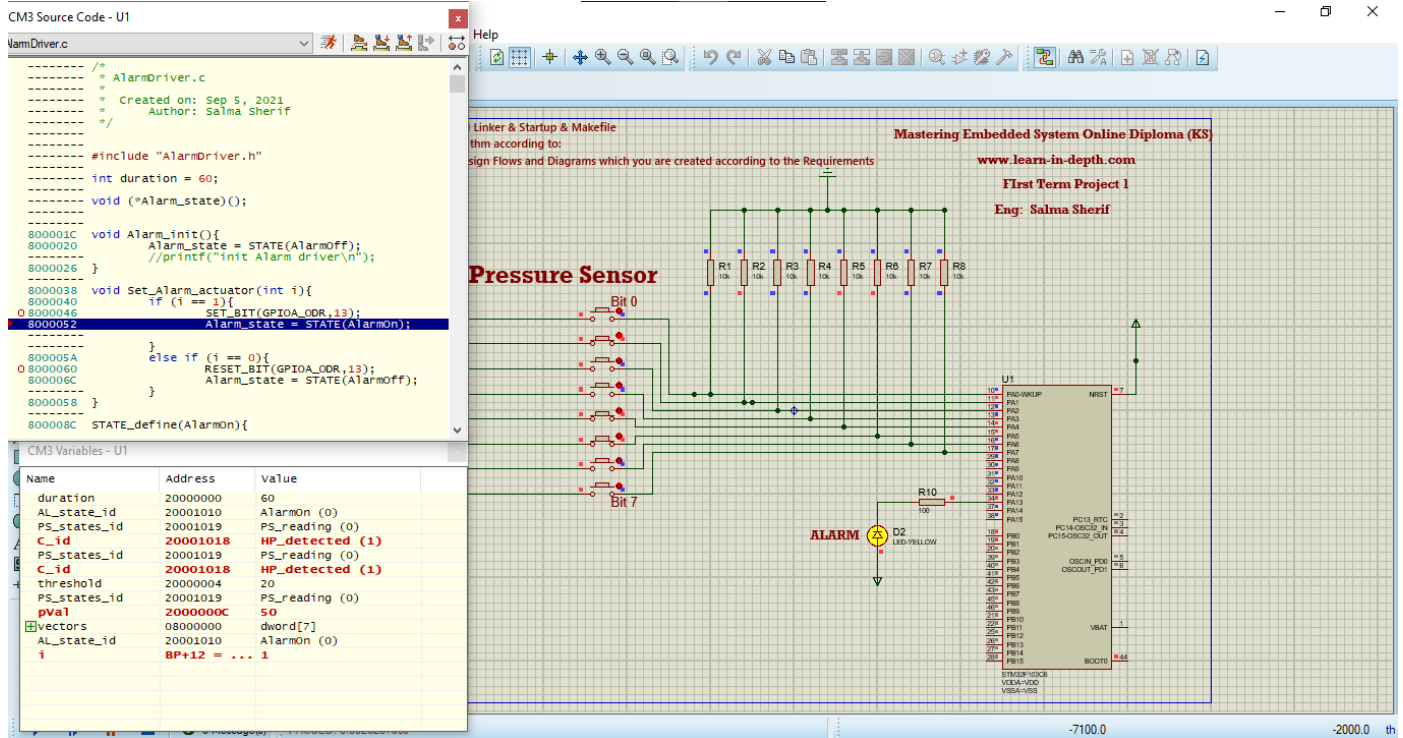
1 > /*...
9 #include "AlarmDriver.h"
10 int duration = 60;
11 void (*Alarm_state)();
12
13 void Alarm_init(){
14     Alarm_state = STATE(AlarmOff);
15     //printf("init Alarm driver\n");
16 }
17
18 void Set_Alarm_actuator(int i){
19     if (i == 1){
20         SET_BIT(GPIOA_ODR,13);
21         Alarm_state = STATE(AlarmOn);
22     }else if (i == 0){
23         RESET_BIT(GPIOA_ODR,13);
24         Alarm_state = STATE(AlarmOff);
25     }
26 }
27
28 STATE_define(AlarmOn){
29     // printf("Alarm is ONNNNNN!!\n");
30     AL_state_id = AlarmOn;
31     // printf("Delaaaaay!!!\n");
32     Delay(duration*1000); //delay 60 sec
33     Alarm_state = STATE(AlarmOff);
34 }
35
36 STATE_define(AlarmOff){
37     //printf("Alarm is OFF!!\n");
38     AL_state_id = AlarmOff;
39 }
  
```

```

1 /*
2  * AlarmDriver.h
3  *
4  * Created on: Sep 5, 2021
5  * Author: Salma Sherif
6  */
7
8 #ifndef ALARMDRIVER_H_
9 #define ALARMDRIVER_H_
10
11 #include "driver.h"
12 #include "states.h"
13
14
15 enum{
16     AlarmOn,
17     AlarmOff
18 }AL_state_id;
19
20 void Alarm_init();
21 STATE_define(AlarmOn);
22 STATE_define(AlarmOff);
23
24 extern void (*Alarm_state)();
25
26 #endif /* ALARMDRIVER_H_ */
27
28
29
30
  
```



The measure pressure was 50 which is greater than 20, so the alarm is ON for 60 seconds



After the 60 seconds ended, the Alarm is turned Off, and would turn on again if the pressure is greater than 20.

CM3 Source Code - U1

```

/* AlarmDriver.c
 * Created on: Sep 5, 2021
 * Author: Salma Sherif
 */

#include "AlarmDriver.h"

int duration = 60;

void (*Alarm_state)();

void Alarm_init(){
    Alarm_state = STATE(AlarmOff);
    //printf("init Alarm driver\n");
}

void Set_Alarm_actuator(int i){
    if (i == 1){
        SET_BIT(GPIOA_ODR, 13);
        Alarm_state = STATE(AlarmOn);
    }
    else if (i == 0){
        RESET_BIT(GPIOA_ODR, 13);
        Alarm_state = STATE(AlarmOff);
    }
}

STATE_define(AlarmOn){
    duration = 60;
    AL_state_id = 20001010;
    PS_states_id = 20001019;
    C_id = 20001018;
    PS_states_id = 20001019;
    C_id = 20001018;
    threshold = 20000004;
    PS_states_id = 20001019;
    pval = 2000000C;
    vectors = 08000000;
    AL_state_id = 20001010;
    BP+12 = ... 0
        
```

CM3 Variables - U1

Name	Address	Value
duration	20000000	60
AL_state_id	20001010	AlarmOn (0)
PS_states_id	20001019	PS_reading (0)
C_id	20001018	HP_detected (1)
PS_states_id	20001019	PS_reading (0)
C_id	20001018	HP_detected (1)
threshold	20000004	20
PS_states_id	20001019	PS_reading (0)
pval	2000000C	50
vectors	08000000	dword[7]
AL_state_id	20001010	AlarmOn (0)
1	BP+12 = ... 0	

Linker & Startup & Makefile  
 thm according to:  
 sign Flows and Diagrams which you are created according to the Requirements

Mastering Embedded System Online Diploma (KS)  
[www.learn-in-depth.com](http://www.learn-in-depth.com)  
**First Term Project 1**  
 Eng: Salma Sherif