FIRST TERM
PROJECT 1
PRESSURE CONTROLLER

ENG : Hazem Mohamed ELNashar

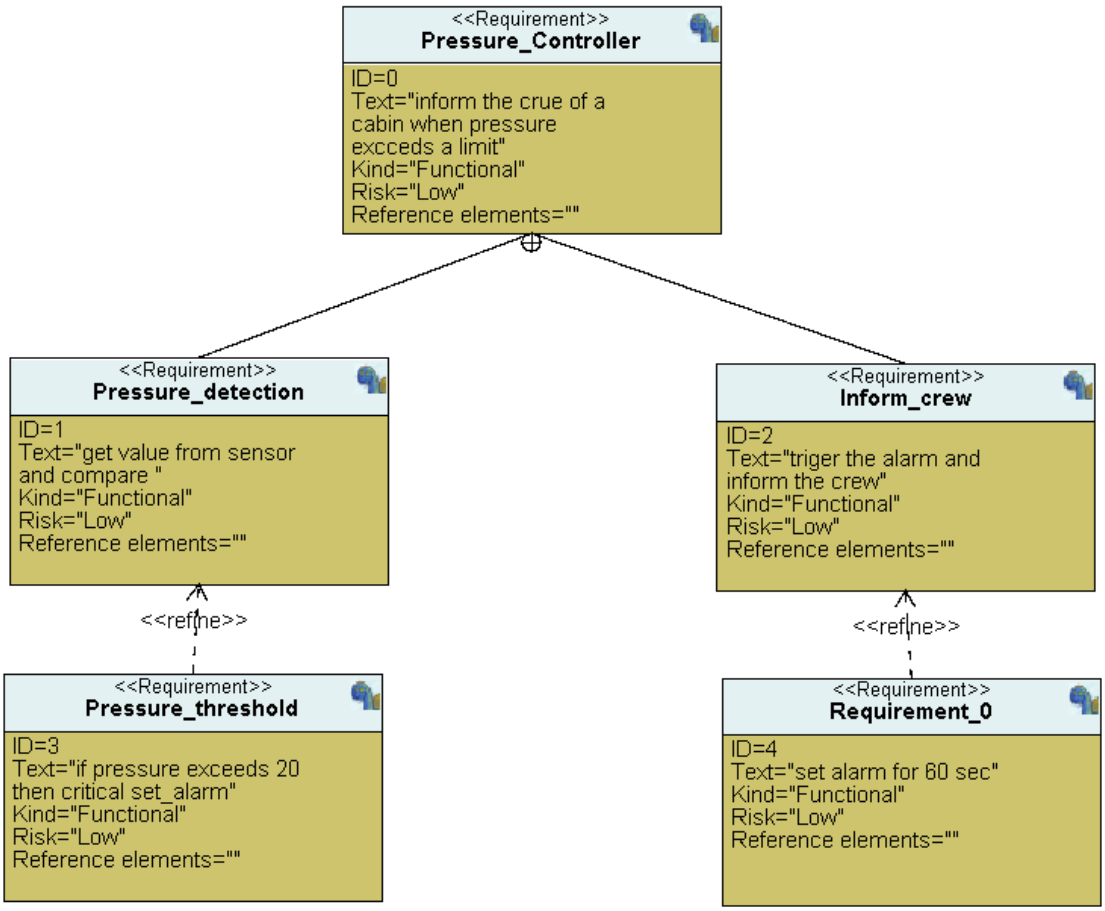
My Profile :
https://www.learn-in-depth.com/online-diploma/hazemmohamed30010@gmail.com

GitHub Repository:
https://github.com/hazem31/Embedded-Assignments

CASE STUDY:

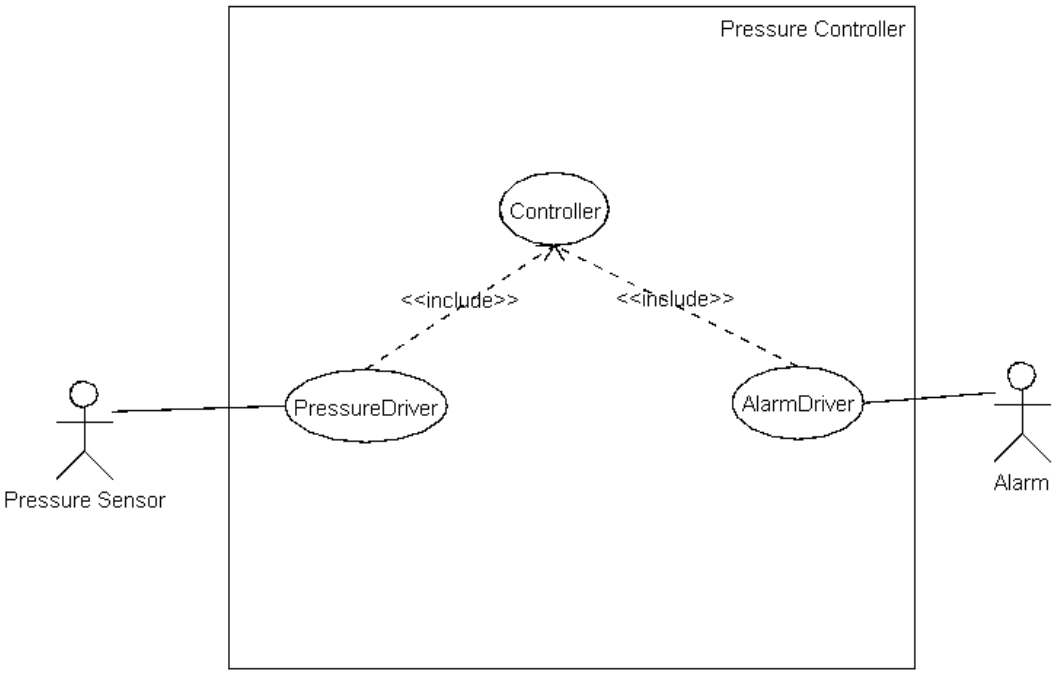
- A pressure controller informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin
- The alarm duration equals 60 seconds

REQUIREMENTS DIAGRAM :

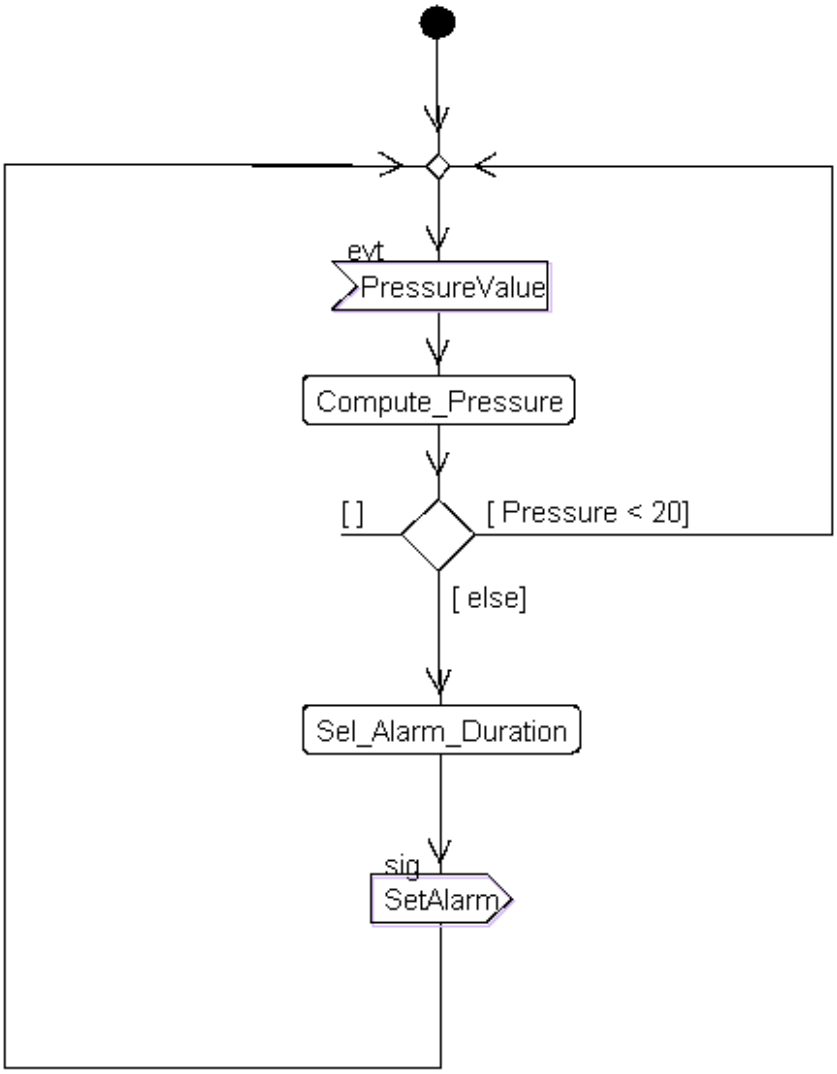


SYSTEM ANALYSIS:

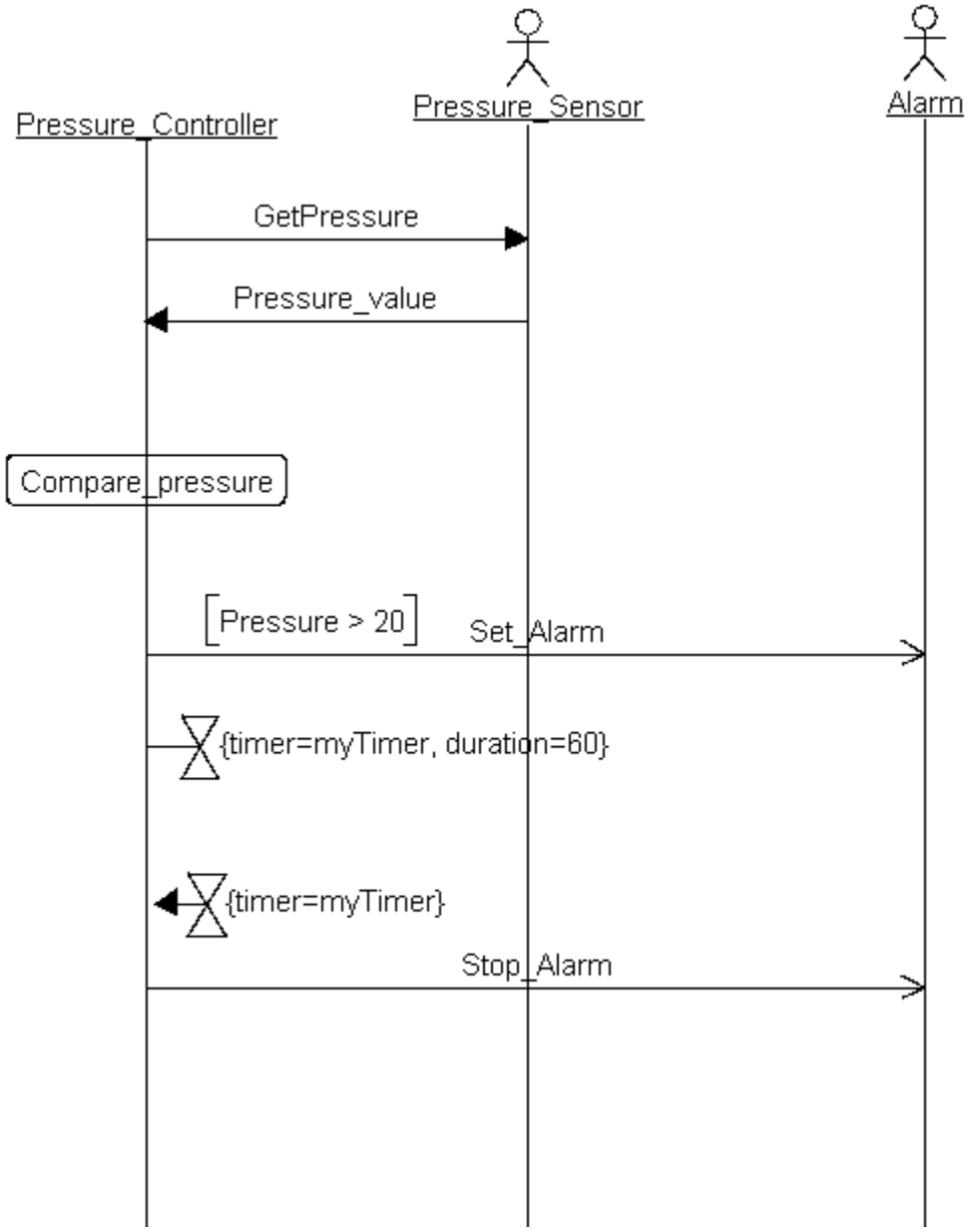
Use Case:


Pressure Controller
Controller
<<include>>
<<include>>
PressureDriver
AlarmDriver
Pressure Sensor
Alarm

Activity Diagram:

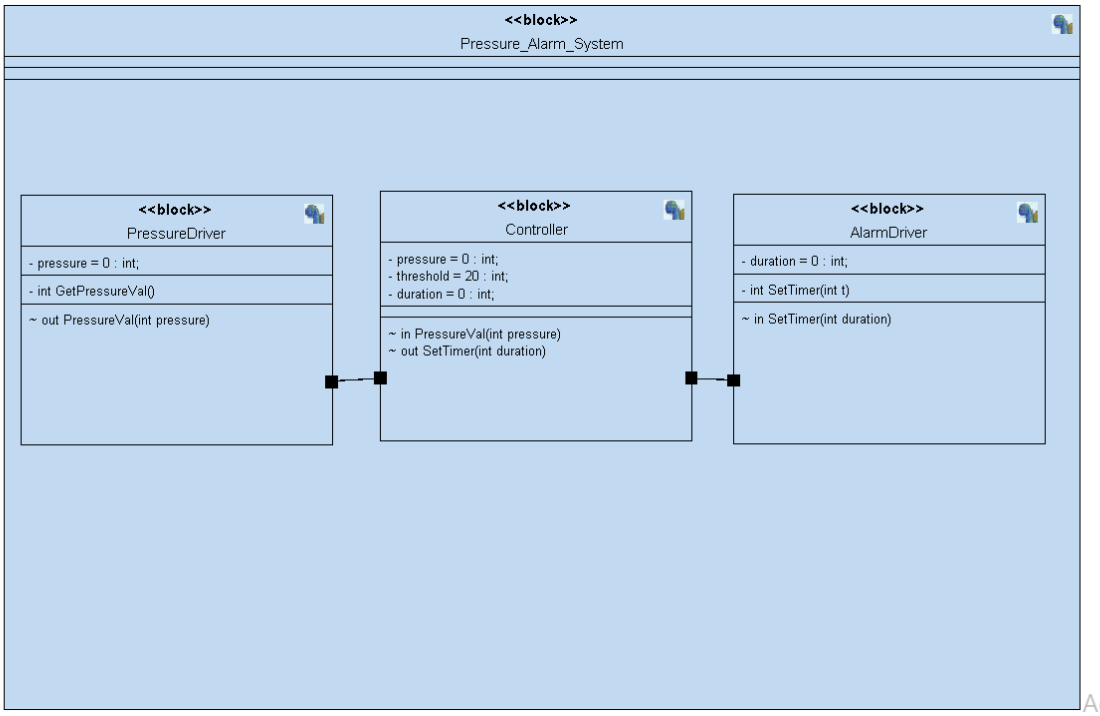


Sequence Diagram:

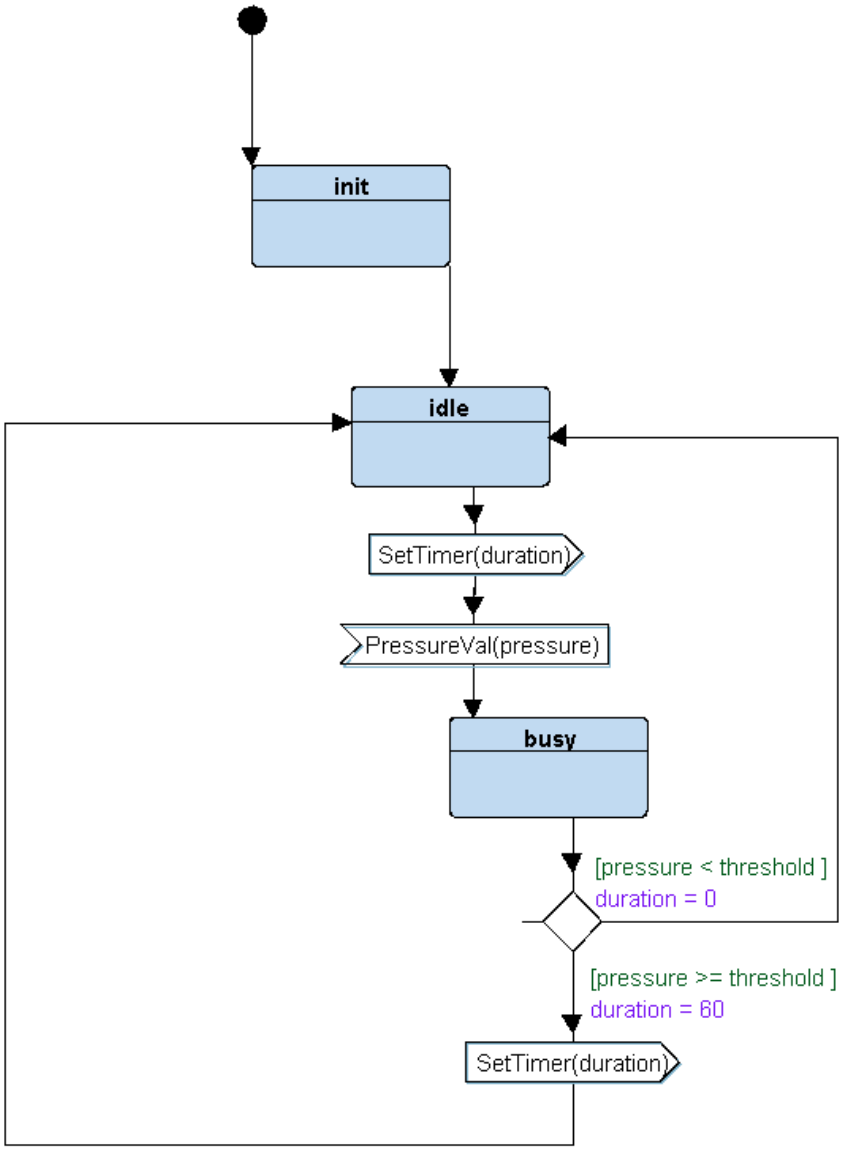


SYSTEM DESIGN:

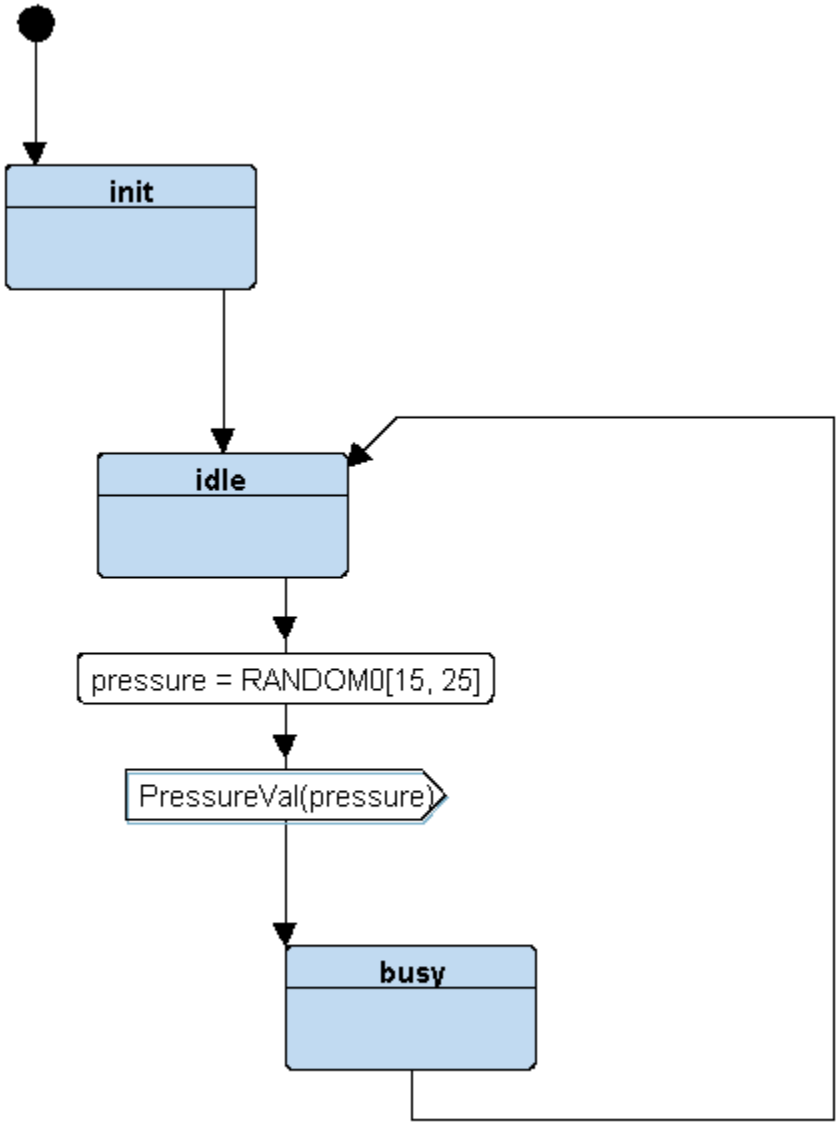
Block Diagram:



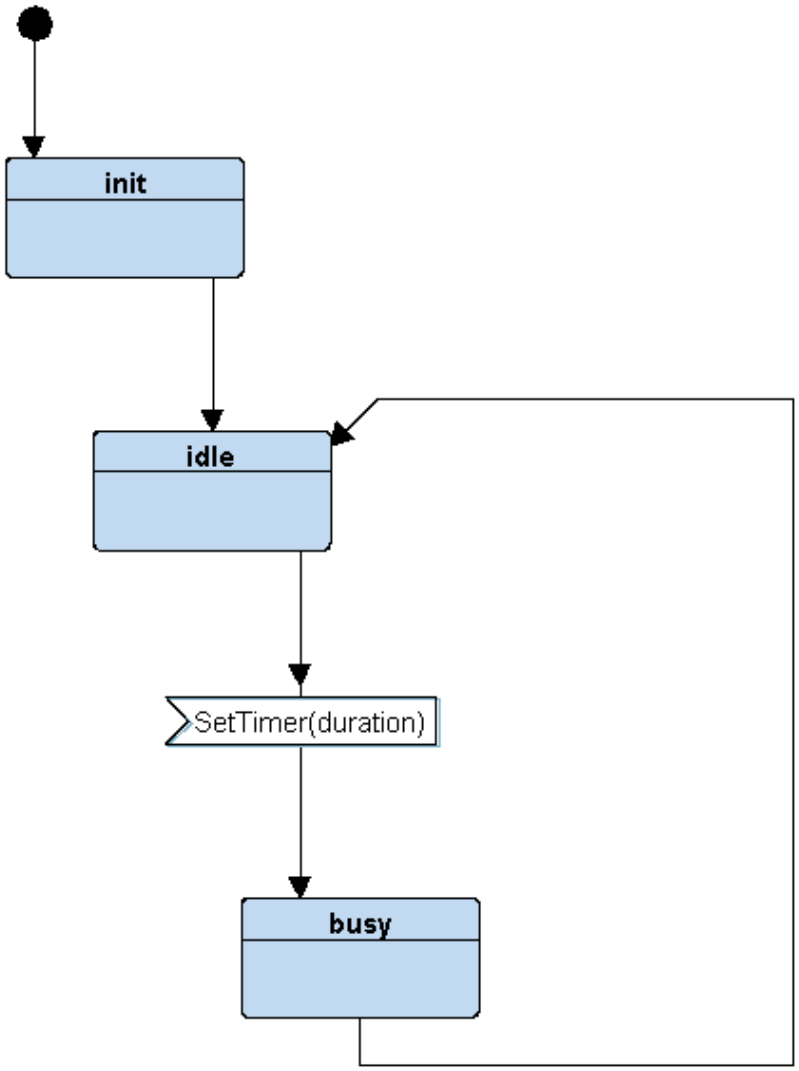
Controller State Diagram:



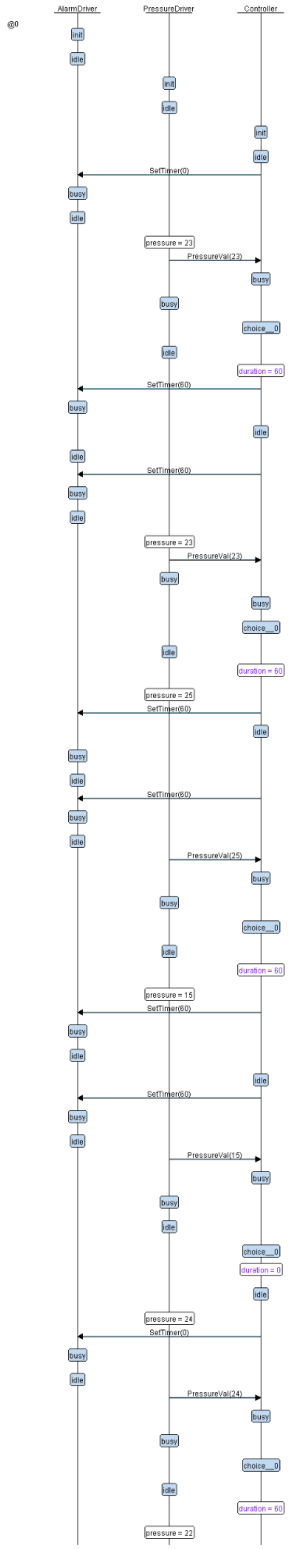
PressureDriver State Diagram:



AlarmDriver State Diagram:



Simulation:


AlarmDriver
PressureDriver
Controller
@0
init
idle
SetTimer(0)
busy
pressure = 23
PressureVal(23)
choice__0
duration = 60
SetTimer(60)
pressure = 25
PressureVal(25)
pressure = 15
PressureVal(15)
duration = 0
pressure = 24
PressureVal(24)
pressure = 22

CODE :

Main.c file :

```
4   #include "driver.h"
5   #include "Sensor.h"
6   #include "Alarm.h"
7   #include "Controller.h"
8
9   int main (){
10      GPIO_INITIALIZATION();
11      PC_state = STATE(PC_idle);
12      AL_state = STATE(AL_idle);
13      PS_state = STATE(PS_idle);
14      while (1)
15      {
16          PS_state();
17          PC_state();
18          AL_state();
19      }
20
21  }
22
```

Controller Files:

```c
#ifndef __CONTROLLER_H_
#define __CONTROLLER_H_

#include"state.h"


enum {
    PC_idle,
    PC_busy
}PC_state_id;

extern void (*PC_state)();

STATE_define(PC_idle);
STATE_define(PC_busy);

#endif
```

```c
#include"Controller.h"


static unsigned int pressure = 0;
static unsigned int duration = 60;
static unsigned int threshold = 20;

void (*PC_state)();

STATE_define(PC_idle) {

    PC_state_id = PC_idle;
    SetTimer(1);
}

STATE_define(PC_busy) {

    PC_state_id = PC_busy;
    if (pressure >= threshold)
    {
        SetTimer(0);
    }
    PC_state = STATE(PC_idle);


}

void PressureVal(int p) {

    pressure = p;
    PC_state = STATE(PC_busy);
}
```

Sensor Files:

```
1
2   #ifndef _SENSOR_H_
3   #define _SENSOR_H_
4
5
6   #include "state.h"
7   #include "driver.h"
8
9
10
11  enum {
12      PS_idle,
13      PS_busy
14  }PS_state_id;
15
16
17  extern void (*PS_state)();
18
19  STATE_define(PS_idle);
20  STATE_define(PS_busy);
21
22
23
24  #endif
```

```
2
3   #include "Sensor.h"
4
5   static unsigned int pressure = 0;
6
7
8   void (*PS_state)();
9
10  STATE_define(PS_idle) {
11
12      PS_state_id = PS_idle;
13
14      pressure = getPressureVal();
15
16      PressureVal(pressure);
17      PS_state = STATE(PS_busy);
18  }
19
20  STATE_define(PS_busy) {
21
22      PS_state_id = PS_busy;
23
24      Delay(1000);
25
26      PS_state = STATE(PS_idle);
27  }
```

Alarm Files:

```c
#ifndef _ALARM_H_
#define _ALARM_H_

#include "state.h"
#include "driver.h"



enum {
    AL_idle,
    AL_busy
}AL_state_id;


extern void (*AL_state)();

STATE_define(AL_idle);
STATE_define(AL_busy);



#endif
```

```c
#include "Alarm.h"

static unsigned int duration = 0;

void (*AL_state)();

STATE_define(AL_idle) {

    AL_state_id = AL_idle;
    duration = 0;
    Set_Alarm_actuator(1);


}

void SetTimer(int t) {

    duration = t;
    AL_state = STATE(AL_busy);
}

STATE_define(AL_busy) {

    AL_state_id = AL_busy;
    Set_Alarm_actuator(duration);
    if (duration == 0)
    {
        Delay(1000000);
        duration = 1;
    }
    AL_state = STATE(AL_idle);


}

```

Driver Files:

```c
#include <stdint.h>
#include <stdio.h>

#define SET_BIT(ADDRESS,BIT)   ADDRESS |=  (1<<BIT)
#define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
#define TOGGLE_BIT(ADDRESS,BIT)  ADDRESS ^=  (1<<BIT)
#define READ_BIT(ADDRESS,BIT) ((ADDRESS) &   (1<<(BIT)))


#define GPIO_PORTA 0x40010800
#define BASE_RCC   0x40021000

#define APB2ENR   *(volatile uint32_t *)(BASE_RCC + 0x18)

#define GPIOA_CRL *(volatile uint32_t *)(GPIO_PORTA + 0x00)
#define GPIOA_CRH *(volatile uint32_t *)(GPIO_PORTA + 0X04)
#define GPIOA_IDR *(volatile uint32_t *)(GPIO_PORTA + 0x08)
#define GPIOA_ODR *(volatile uint32_t *)(GPIO_PORTA + 0x0C)


void Delay(int nCount);
int getPressureVal();
void Set_Alarm_actuator(int i);
void GPIO_INITIALIZATION ();

```

```c
#include "driver.h"
#include <stdint.h>
#include <stdio.h>
void Delay(int nCount)
{
    for(; nCount != 0; nCount--);
}

int getPressureVal(){
    return (GPIOA_IDR & 0xFF);
}

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

void GPIO_INITIALIZATION (){
    SET_BIT(APB2ENR, 2);
    GPIOA_CRL &= 0xFF0FFFFF;
    GPIOA_CRL |= 0x00000000;
    GPIOA_CRH &= 0xFF0FFFFF;
    GPIOA_CRH |= 0x22222222;
}

```

SYMBOLS TABLE

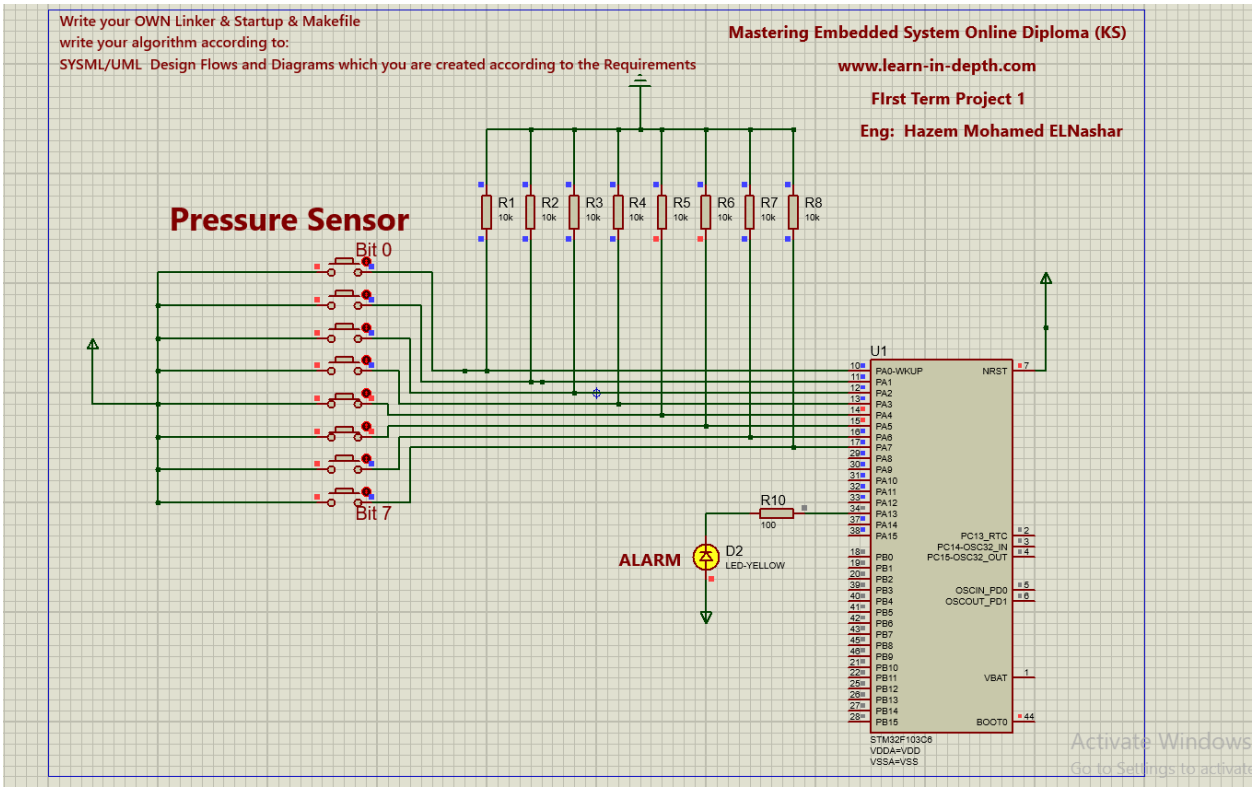
```
20000014 B _E_bss
20000008 D _E_DATA
0800036c T _E_text
20000008 B _S_bss
20000000 D _S_DATA
20001014 B _stack_top
20001018 B AL_state
20001014 B AL_state_id
08000354 T Bus_Fault
08000134 T Delay
20000000 d duration
20000008 b duration
08000154 T getPressureVal
080001a8 T GPIO_INITIALIZATION
0800033c T H_Handler
080001f8 T main
08000348 T MM_Fault_Handler
08000330 T NMI_Handler
2000101c B PC_state
20001020 B PC_state_id
2000000c b pressure
20000010 b pressure
08000108 T PressureVal
20001024 B PS_state
20001021 B PS_state_id
080002a4 T Reset_Handler
0800016c T Set_Alarm_actuator
08000040 T SetTimer
0800006c T ST_AL_busy
0800001c T ST_AL_idle
080000cc T ST_PC_busy
080000b4 T ST_PC_idle
0800027c T ST_PS_busy
08000240 T ST_PS_idle
20000004 d threshold
08000360 T Usage_Fault_Handler
08000000 T vectors
```

Map Files

```
.text           0x08000000      0x36c
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x0800001c       0x98 Alarm.o
                0x0800001c                ST_AL_idle
                0x08000040                SetTimer
                0x0800006c                ST_AL_busy
 .text          0x080000b4       0x80 Controller.o
                0x080000b4                ST_PC_idle
                0x080000cc                ST_PC_busy
                0x08000108                PressureVal
 .text          0x08000134       0xc4 driver.o
                0x08000134                Delay
                0x08000154                getPressureVal
                0x0800016c                Set_Alarm_actuator
                0x080001a8                GPIO_INITIALIZATION
 .text          0x080001f8       0x48 main.o
                0x080001f8                main
 .text          0x08000240       0x64 Sensor.o
                0x08000240                ST_PS_idle
                0x0800027c                ST_PS_busy
 .text          0x080002a4       0xc8 startup.o
                0x080002a4                Reset_Handler
                0x08000330                NMI_Handler
                0x0800033c                H_Handler
                0x08000348                MM_Fault_Handler
                0x08000354                Bus_Fault
                0x08000360                Usage_Fault_Handler
 *(.rodata*)
                0x0800036c                _E_text = .

 .glue_7        0x0800036c        0x0
  .glue_7       0x0800036c        0x0 linker stubs
```

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: Hazem Mohamed ELNashar
Pressure Sensor
Bit 0
Bit 7
R1 10k
R2 10k
R3 10k
R4 10k
R5 10k
R6 10k
R7 10k
R8 10k
R10 100
ALARM
D2 LED-YELLOW
U1
STM32F103C6
VDDA=VDD
VSSA=VSS
Activate Windows
Go to Settings to activate