Mastering Embedded Systems Online Diploma

www.learn-in-depth.com

First Term (Final Project 1)

Eng. Mostafa Mohamed Elsayed Elsayed Emary

My Profile:

https://www.learn-in-depth.com/online-diploma/mostafa.emary255%40gmail.com/

Abstract: in this report project as the requirement is to make a pressure detection alarm, first it's important to make our case study: meeting with the customer and take his requirements take caring of some assumptions. Then in this report the V-CYCLE is used as model shape. Then make final decisions about requirements table with the customer, pressure sensor on stm32 board is the hardware used to load the SW on it for this project, then the system analysis should be obvious to the customer. then we can implement the system by SW APP, finally the testing and validation tests for the system.

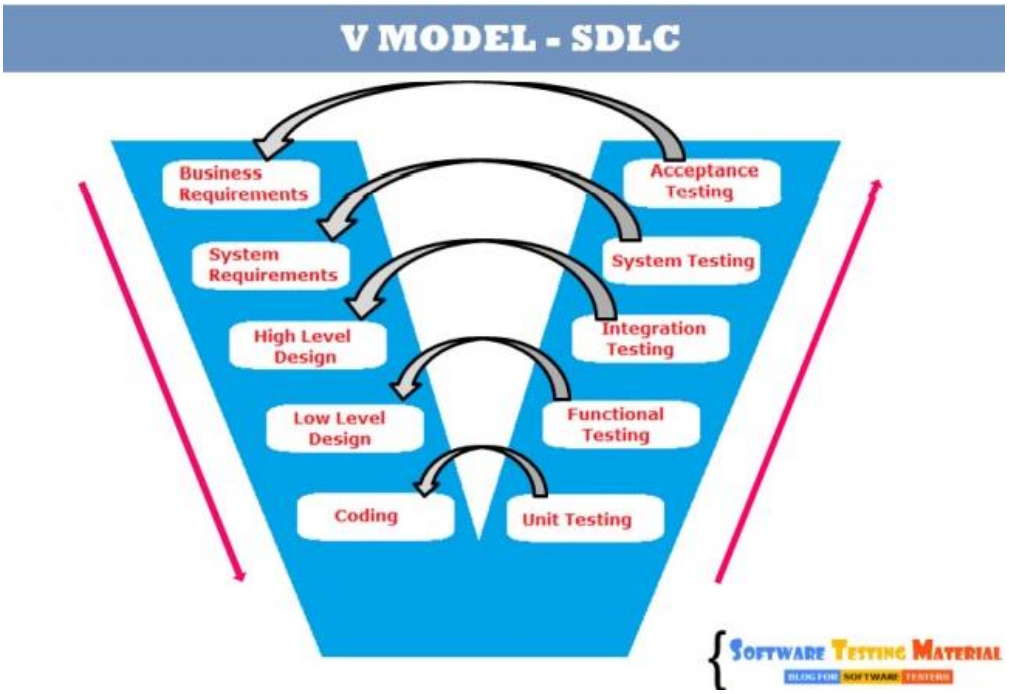
A) customer meeting: informs that he need a SW of the following system.

1- pressure controller detect the pressure value inside the cabin.

2- informs the crew of a cabin with an alarm when the pressure exceeds 20 bar.

3- the alarm duration equals 60 seconds.

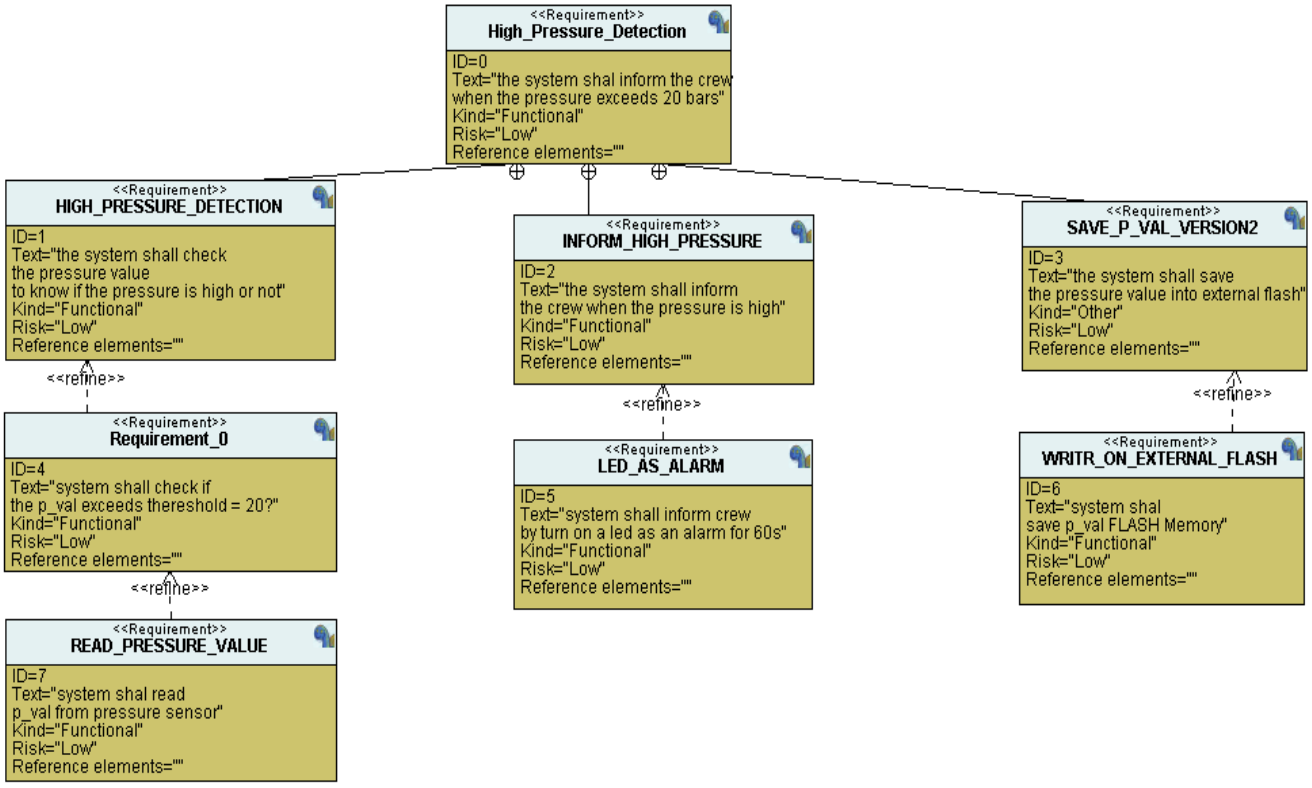
B) Method: V-CYCLE:

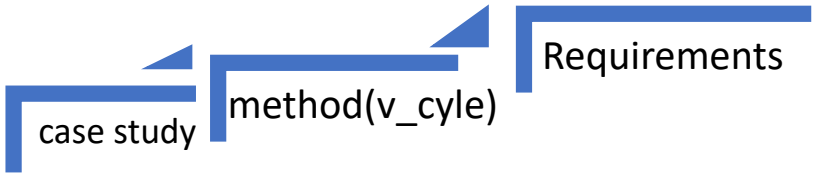


case study

method(v_cyle)

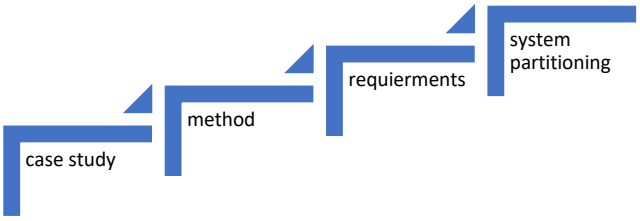
C) REQUIRMENTS ANALYSIS:





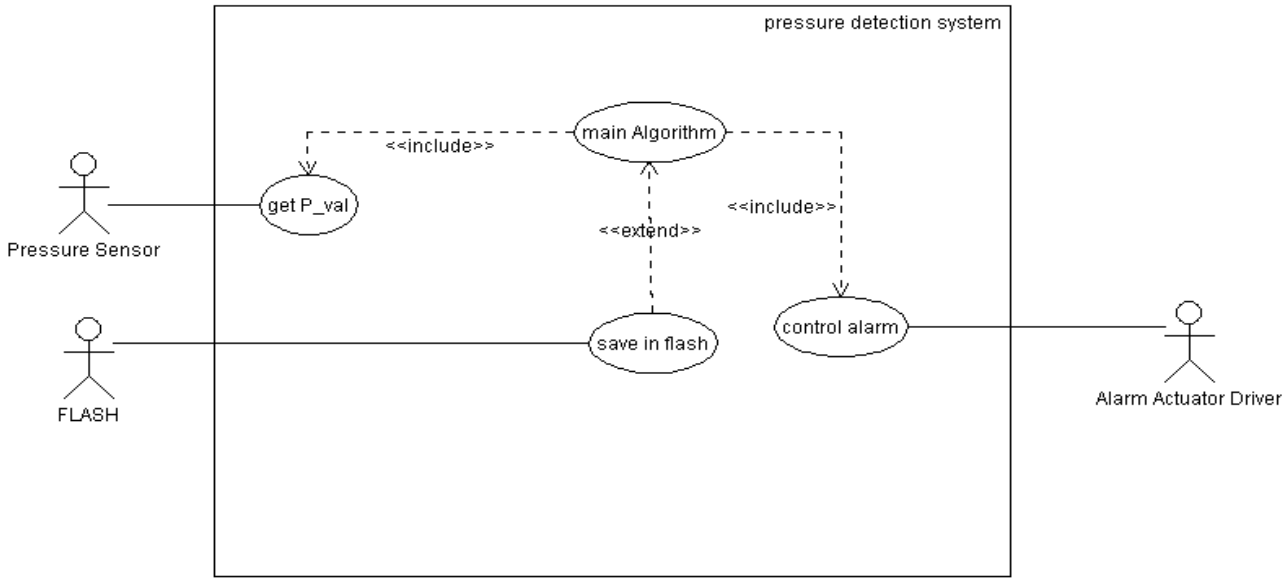
system shall inform the crew when the pressure exceeds 20 bars?	✓
Read pressure value from pressure sensor?	✓
Use a led as an alarm?	✓
Duration time is 60s?	✓
save pressure value FLASH Memory in the next version?	✓

D) S.W: consists of 3 modules will be loaded on stm 32 H.W soc

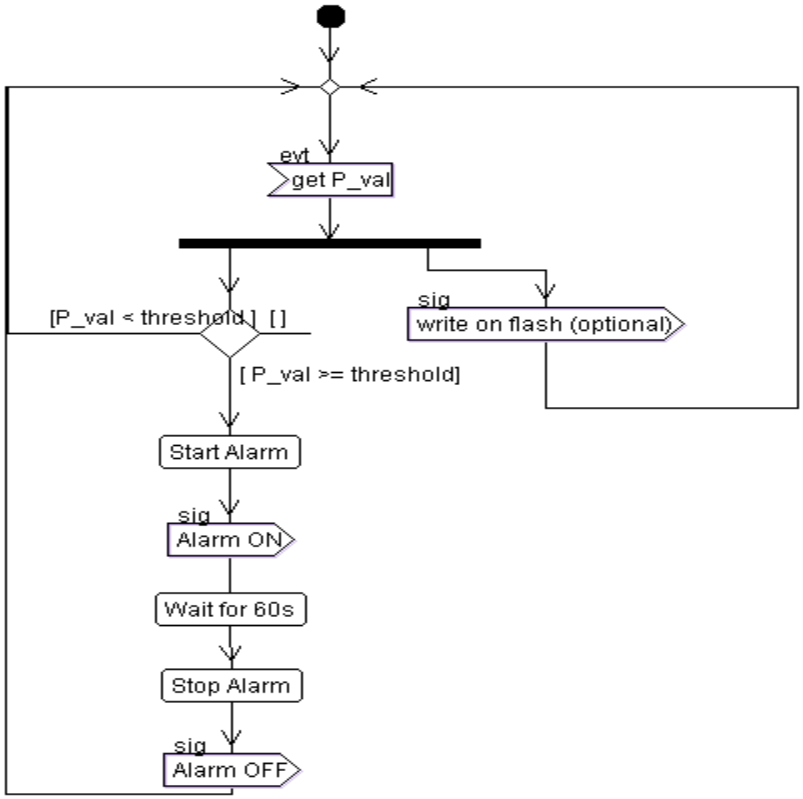


E) system analysis:

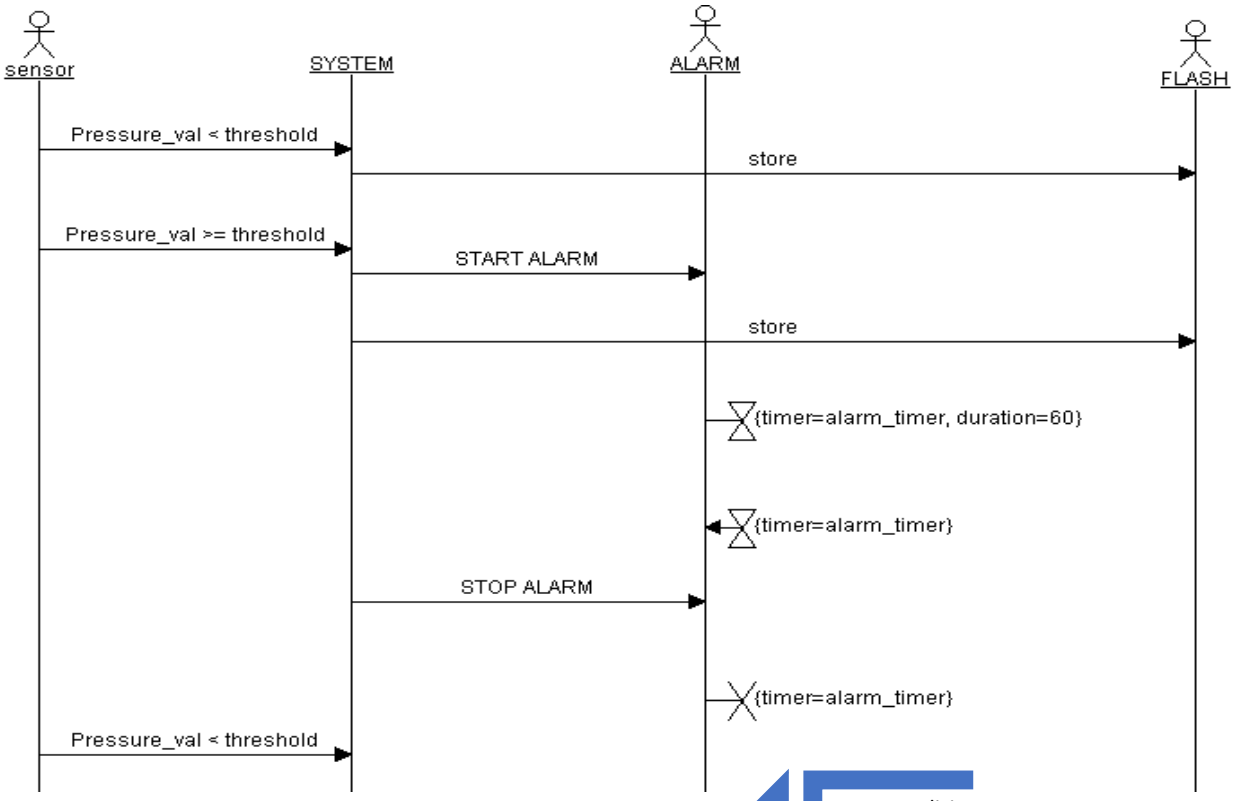
1- USE CASE DIAGRAM: system boundary & main functions content.

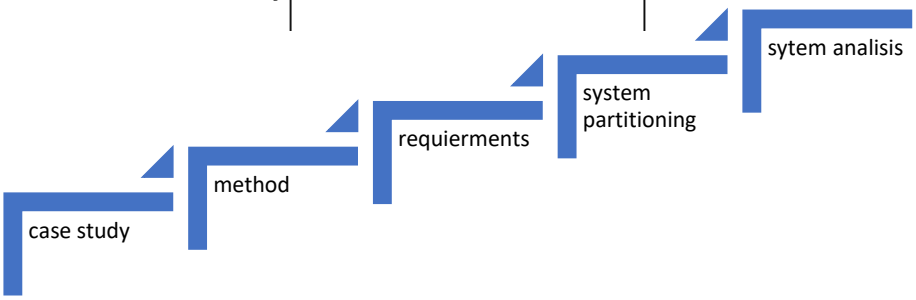


2- ACTIVITY DIAGRAM: relation between the main functions and it's connection between actors . "what main algorithm functions can do."

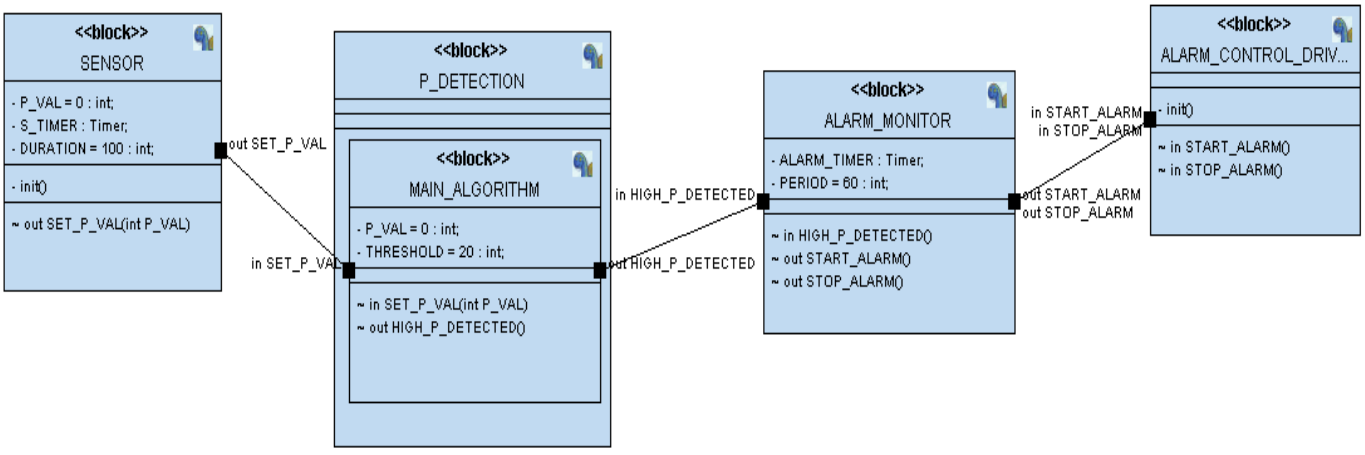


3- SEQUENCE DIAGRAM: communication between system and actors.

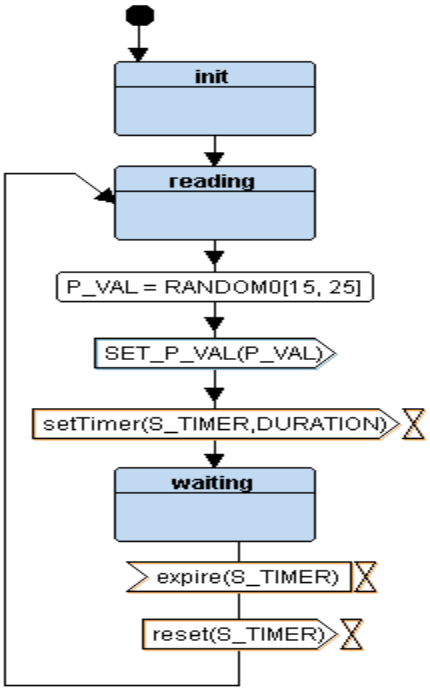




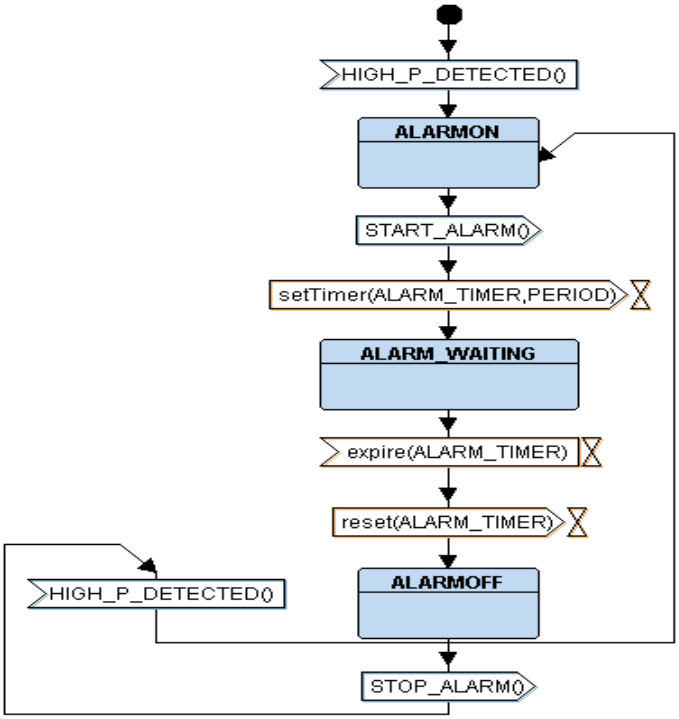
F) SYSTEM DESIGN: BLOCK DIAGRAM.



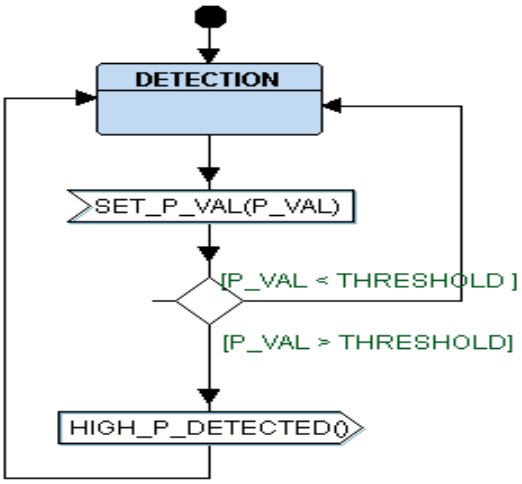
1- SENSOR



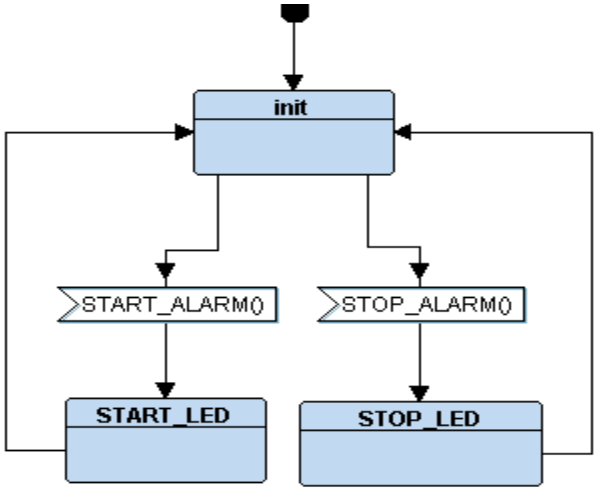
2-ALARM MONIROR

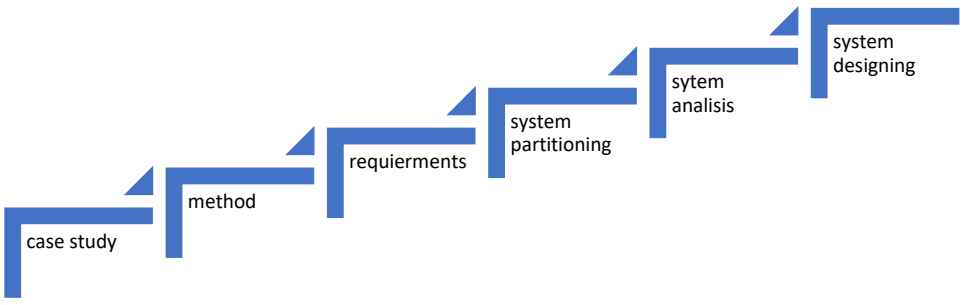


3- MAIN ALGORITHM



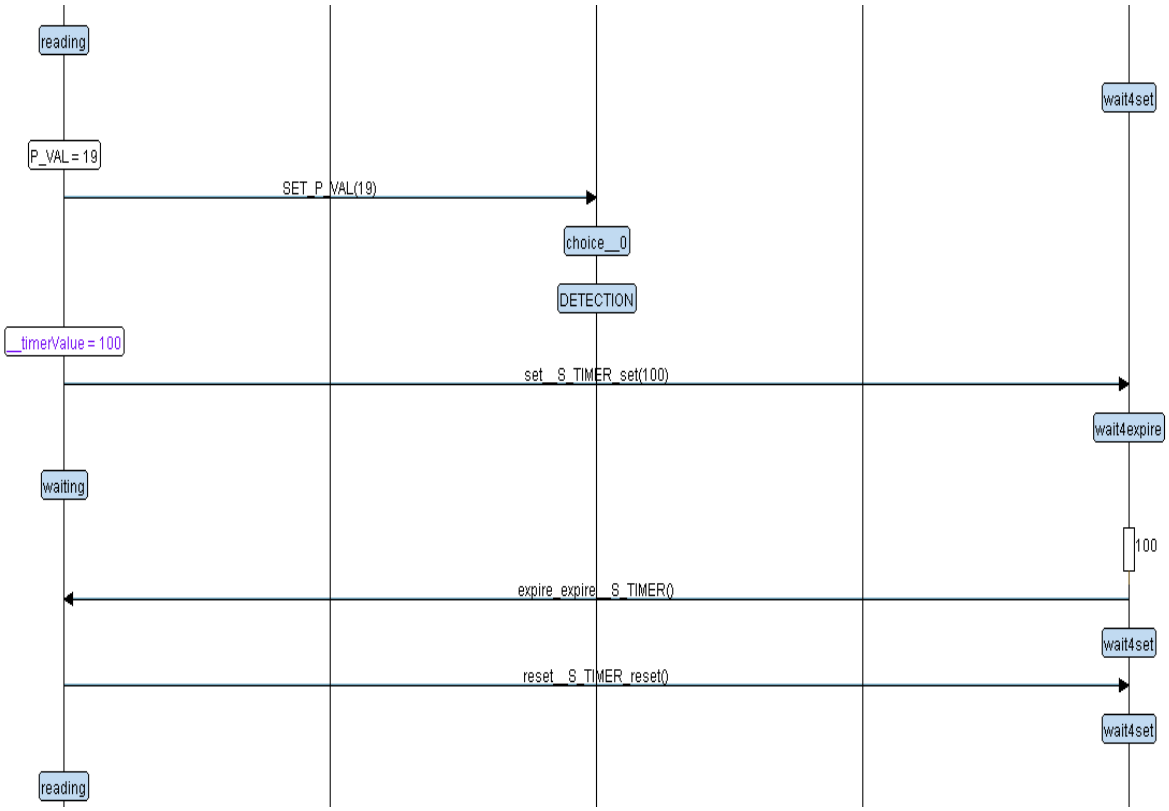
4- ALARM DRIVER

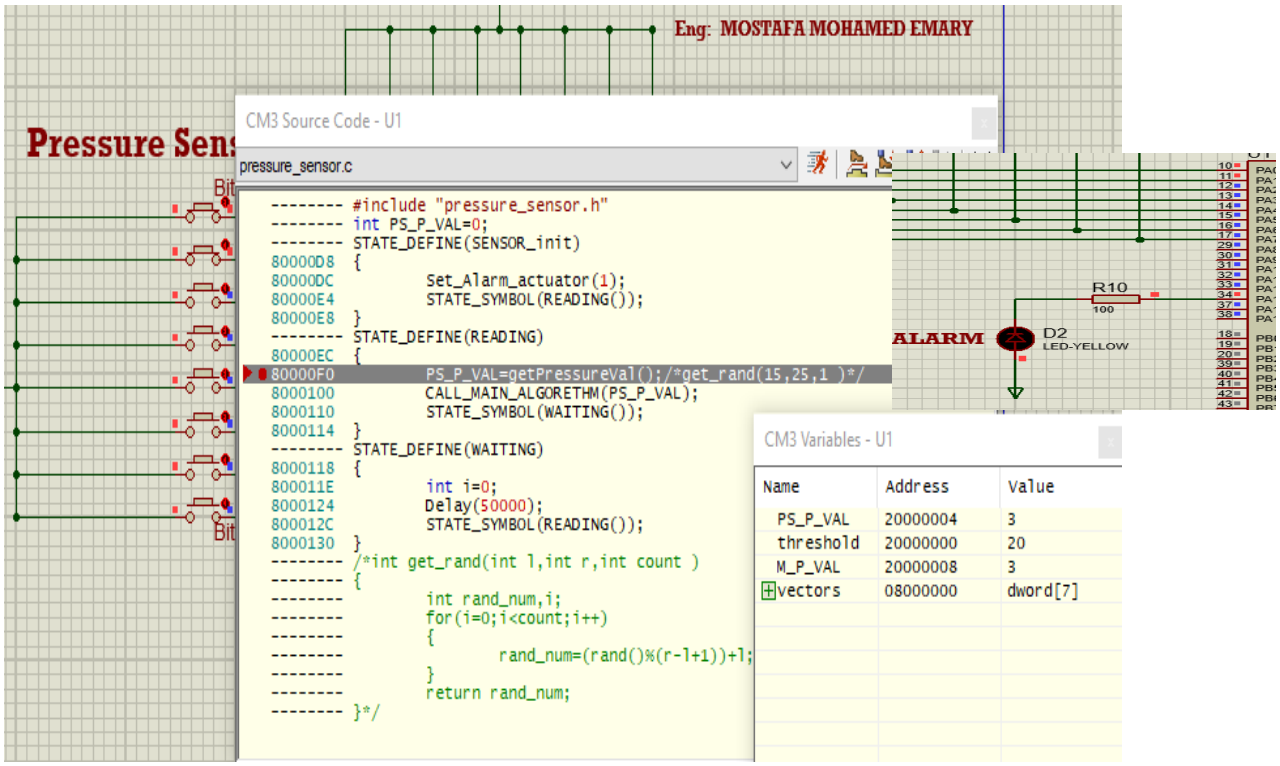


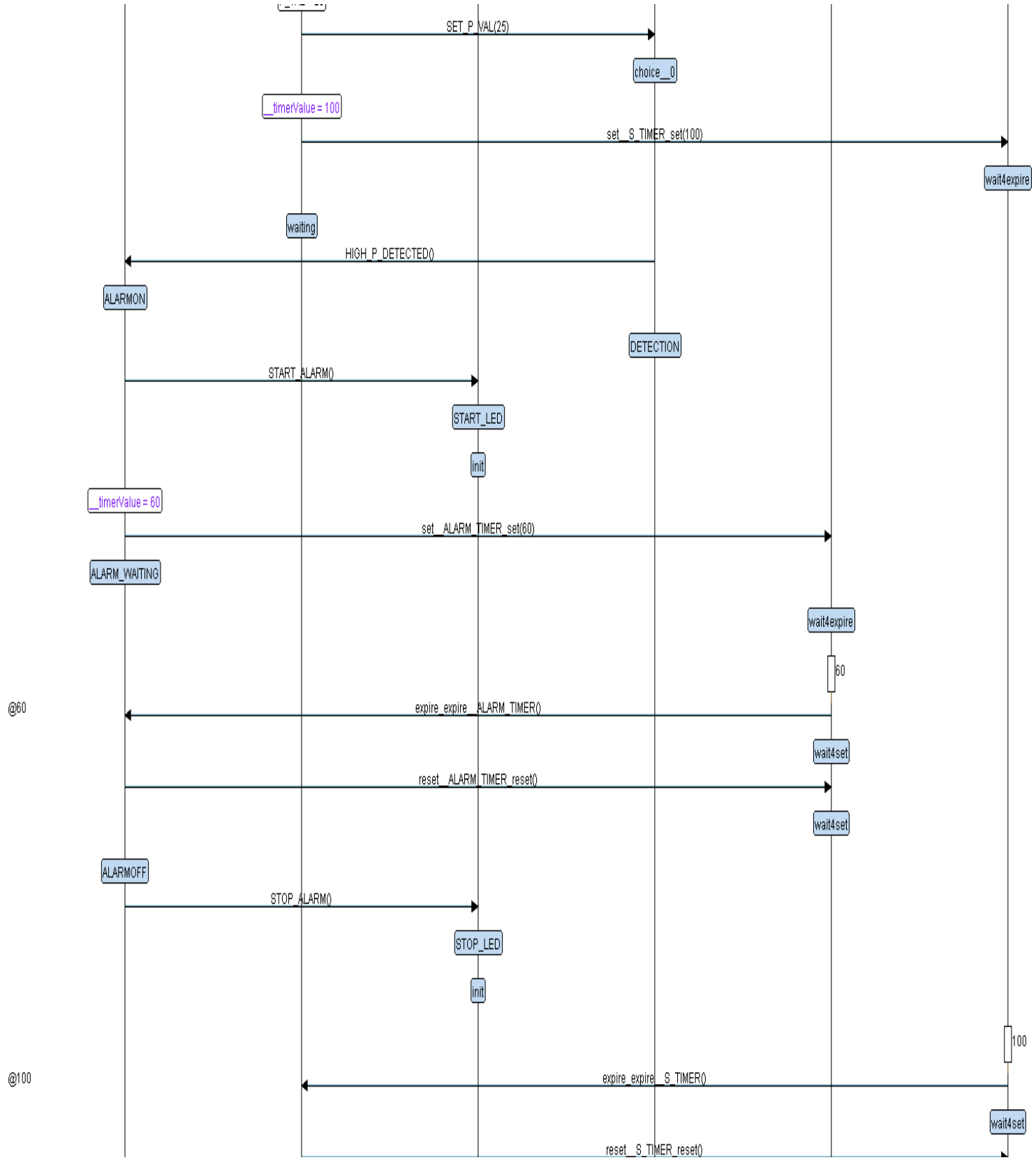


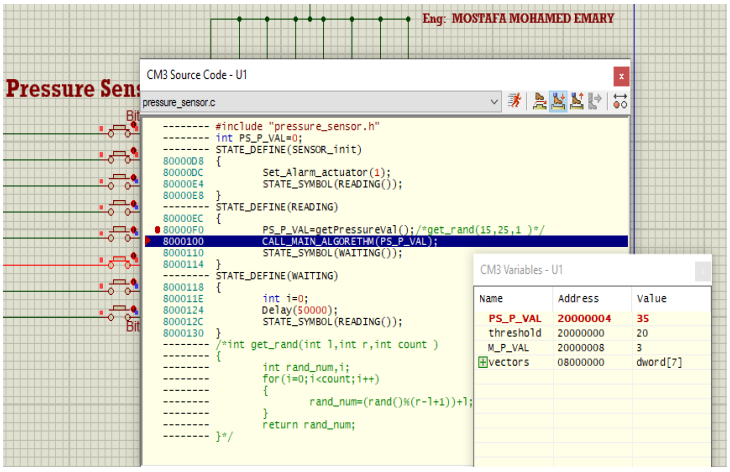
SYSTEM TESTING:

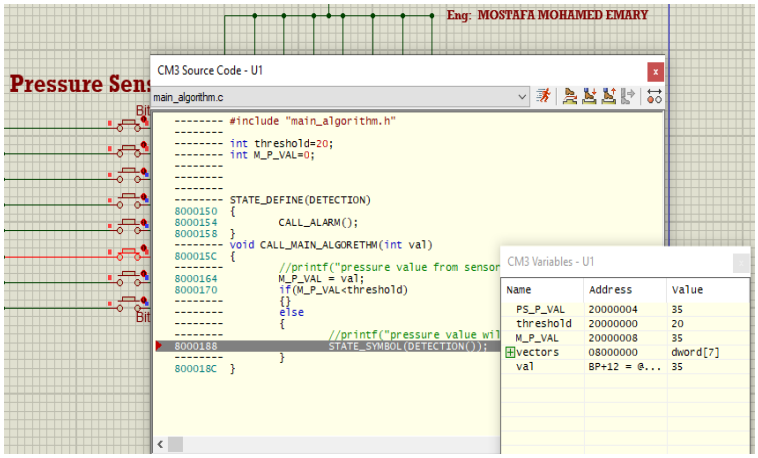
1st case if pressure < threshold:

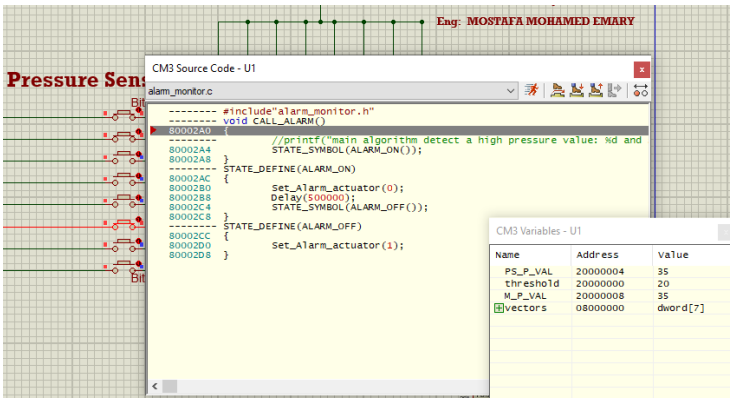


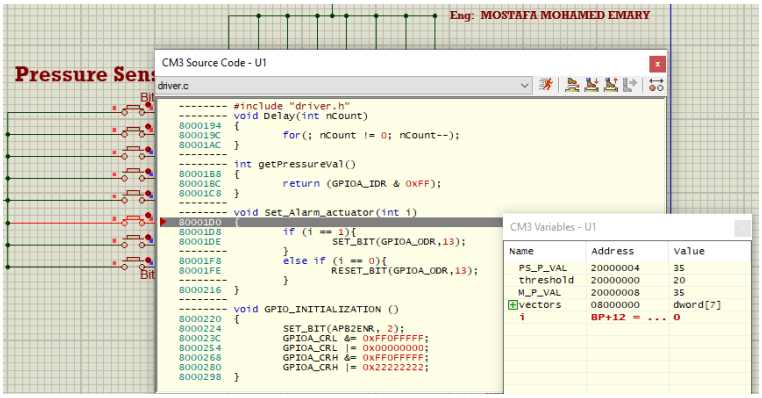


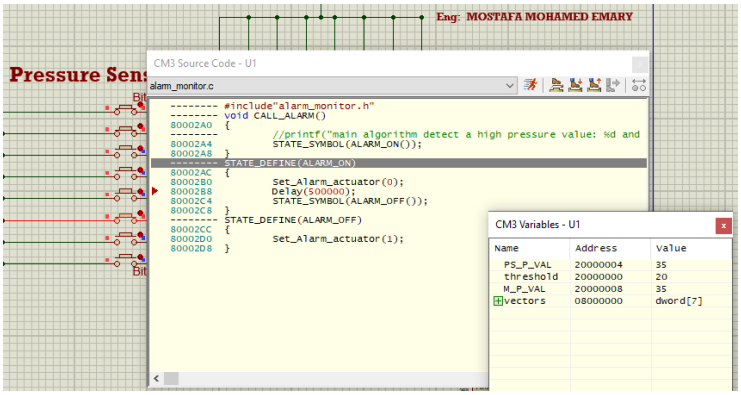

SET_P_VAL(25)
choice__0
__timerValue = 100
set__S_TIMER_set(100)
wait4expire
waiting
HIGH_P_DETECTED()
ALARMON
DETECTION
START_ALARM()
START_LED
init
__timerValue = 60
set__ALARM_TIMER_set(60)
ALARM_WAITING
wait4expire
60
@60
expire_expire__ALARM_TIMER()
wait4set
reset__ALARM_TIMER_reset()
wait4set
ALARMOFF
STOP_ALARM()
STOP_LED
init
100
@100
expire_expire__S_TIMER()
wait4set
reset__S_TIMER_reset()

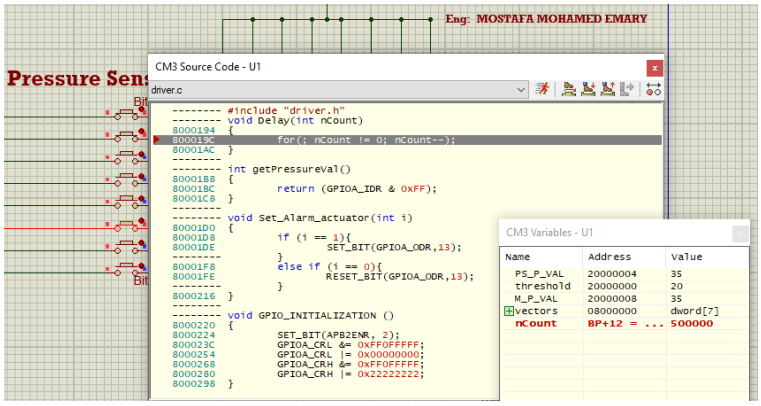

Eng: MOSTAFA MOHAMED EMARY
Pressure Sens
CM3 Source Code - U1
pressure_sensor.c
#include "pressure_sensor.h"
int PS_P_VAL=0;
STATE_DEFINE(SENSOR_init)
Set_Alarm_actuator(1);
STATE_SYMBOL(READING());
STATE_DEFINE(READING)
PS_P_VAL=getPressureVal();/*get_rand(15,25,1)*/
CALL_MAIN_ALGORETHM(PS_P_VAL);
STATE_SYMBOL(WAITING());
STATE_DEFINE(WAITING)
int i=0;
Delay(50000);
STATE_SYMBOL(READING());
/*int get_rand(int l,int r,int count)
int rand_num,i;
for(i=0;i<count;i++)
rand_num=(rand()%(r-l+1))+l;
return rand_num;
}*/
CM3 Variables - U1
Name
Address
Value
PS_P_VAL
20000004
35
threshold
20000000
20
M_P_VAL
20000008
3
vectors
08000000
dword[7]

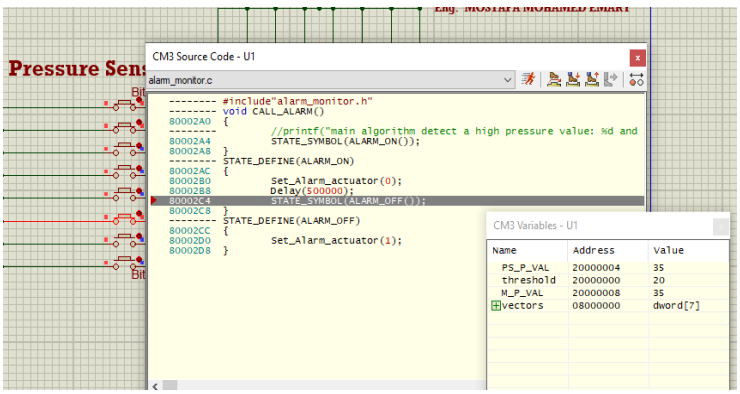

Eng: MOSTAFA MOHAMED EMARY
Pressure Sens
CM3 Source Code - U1
main_algorithm.c
#include "main_algorithm.h"
int threshold=20;
int M_P_VAL=0;
STATE_DEFINE(DETECTION)
CALL_ALARM();
void CALL_MAIN_ALGORETHM(int val)
//printf("pressure value from sensor
M_P_VAL = val;
if(M_P_VAL<threshold)
{}
else
{
//printf("pressure value wil
STATE_SYMBOL(DETECTION());
}
CM3 Variables - U1
Name
Address
Value
PS_P_VAL
20000004
35
threshold
20000000
20
M_P_VAL
20000008
35
vectors
08000000
dword[7]
val
BP+12 = @...
35

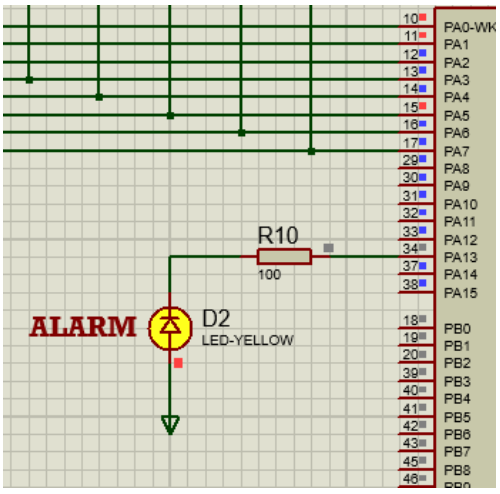

Eng: MOSTAFA MOHAMED EMARY
Pressure Sens
CM3 Source Code - U1
alarm_monitor.c
#include"alarm_monitor.h"
void CALL_ALARM()
//printf("main algorithm detect a high pressure value: %d and
STATE_SYMBOL(ALARM_ON());
STATE_DEFINE(ALARM_ON)
Set_Alarm_actuator(0);
Delay(500000);
STATE_SYMBOL(ALARM_OFF());
STATE_DEFINE(ALARM_OFF)
Set_Alarm_actuator(1);
CM3 Variables - U1
Name
Address
Value
PS_P_VAL
20000004
35
threshold
20000000
20
M_P_VAL
20000008
35
vectors
08000000
dword[7]


Eng: MOSTAFA MOHAMED EMARY
Pressure Sens
CM3 Source Code - U1
driver.c
#include "driver.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;
CM3 Variables - U1
Name
Address
Value
PS_P_VAL
20000004
35
threshold
20000000
20
M_P_VAL
20000008
35
vectors
08000000
dword[7]
i
BP+12 = ...
0


Eng: MOSTAFA MOHAMED EMARY
Pressure Sens
CM3 Source Code - U1
alarm_monitor.c
#include"alarm_monitor.h"
void CALL_ALARM()
//printf("main algorithm detect a high pressure value: %d and
STATE_SYMBOL(ALARM_ON());
STATE_DEFINE(ALARM_ON)
Set_Alarm_actuator(0);
Delay(500000);
STATE_SYMBOL(ALARM_OFF());
STATE_DEFINE(ALARM_OFF)
Set_Alarm_actuator(1);
CM3 Variables - U1
Name
Address
Value
PS_P_VAL
20000004
35
threshold
20000000
20
M_P_VAL
20000008
35
vectors
08000000
dword[7]

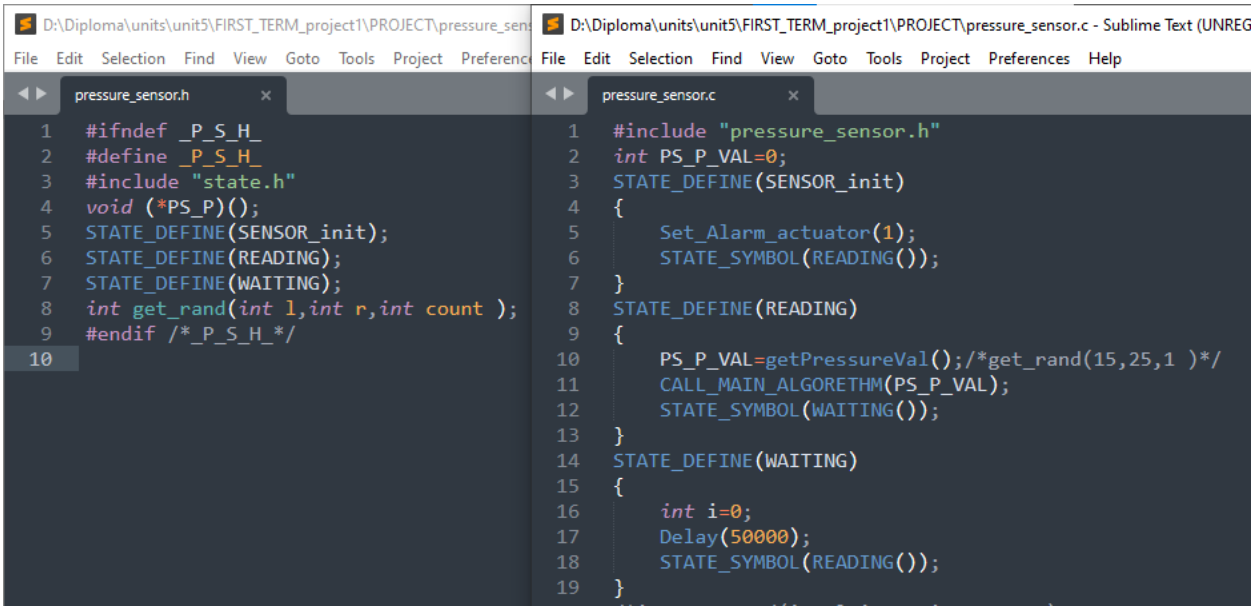

Eng: MOSTAFA MOHAMED EMARY
Pressure Sens
CM3 Source Code - U1
driver.c
#include "driver.h"
void Delay(int nCount)
for(; nCount != 0; nCount--);
int getPressureVal()
return (GPIOA_IDR & 0xFF);
void Set_Alarm_actuator(int i)
if (i == 1){
SET_BIT(GPIOA_ODR,13);
}
else if (i == 0){
RESET_BIT(GPIOA_ODR,13);
}
void GPIO_INITIALIZATION ()
SET_BIT(APB2ENR, 2);
GPIOA_CRL &= 0xFF0FFFFF;
GPIOA_CRL |= 0x00000000;
GPIOA_CRH &= 0xFF0FFFFF;
GPIOA_CRH |= 0x22222222;
CM3 Variables - U1
Name
Address
Value
PS_P_VAL
20000004
35
threshold
20000000
20
M_P_VAL
20000008
35
vectors
08000000
dword[7]
nCount
BP+12 = ...
500000


Pressure Sens
CM3 Source Code - U1
alarm_monitor.c
#include"alarm_monitor.h"
void CALL_ALARM()
//printf("main algorithm detect a high pressure value: %d and
STATE_SYMBOL(ALARM_ON());
STATE_DEFINE(ALARM_ON)
Set_Alarm_actuator(0);
Delay(500000);
STATE_SYMBOL(ALARM_OFF());
STATE_DEFINE(ALARM_OFF)
Set_Alarm_actuator(1);
CM3 Variables - U1
Name
Address
Value
PS_P_VAL
20000004
35
threshold
20000000
20
M_P_VAL
20000008
35
vectors
08000000
dword[7]


PA0-WK
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PB8
R10
100
ALARM
D2
LED-YELLOW

SW.EMPLEMENTATION: 4MODULES

1- PRESSURE SENSOR:



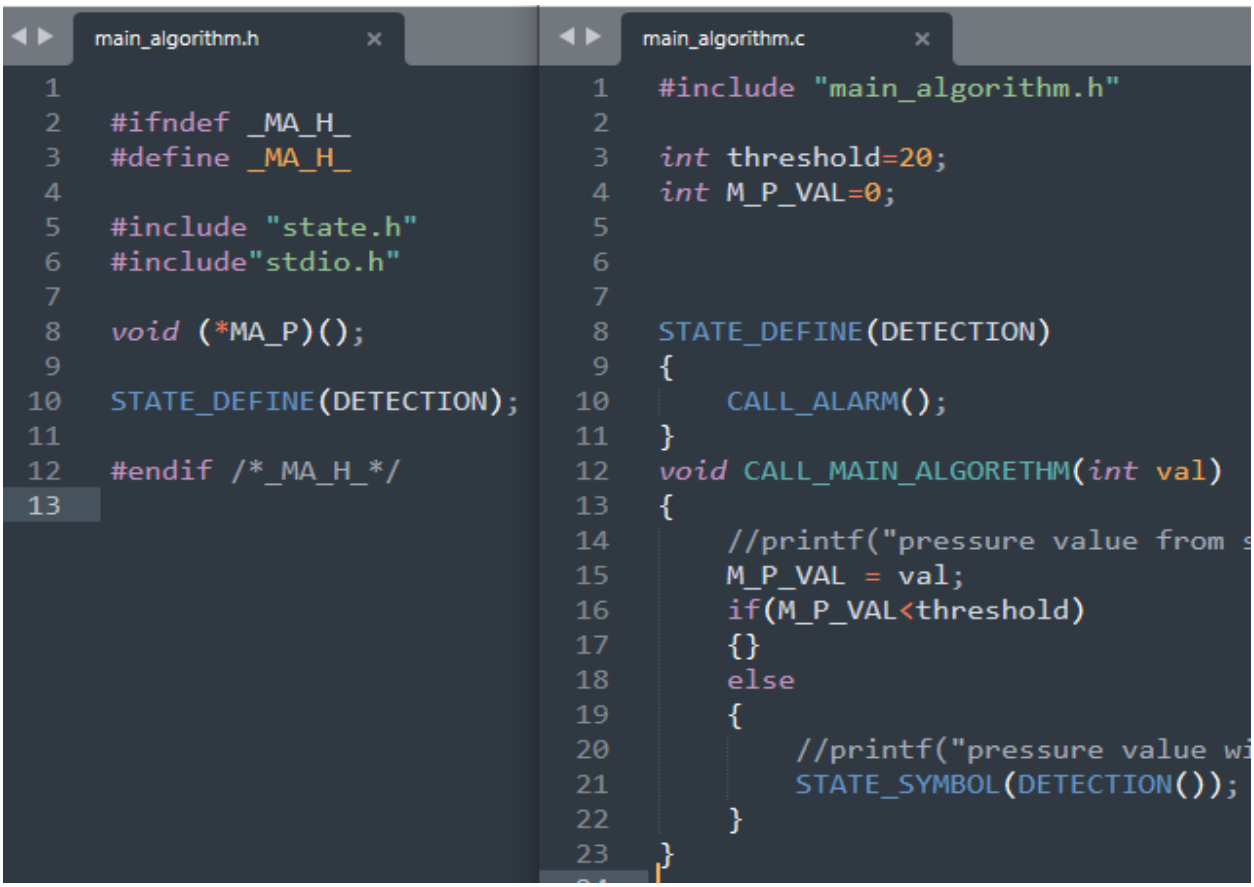
pressure_sensor.h

```
#ifndef _P_S_H_
#define _P_S_H_
#include "state.h"
void (*PS_P)();
STATE_DEFINE(SENSOR_init);
STATE_DEFINE(READING);
STATE_DEFINE(WAITING);
int get_rand(int l,int r,int count );
#endif /*_P_S_H_*/
```

pressure_sensor.c

```
#include "pressure_sensor.h"
int PS_P_VAL=0;
STATE_DEFINE(SENSOR_init)
{
    Set_Alarm_actuator(1);
    STATE_SYMBOL(READING());
}
STATE_DEFINE(READING)
{
    PS_P_VAL=getPressureVal();/*get_rand(15,25,1 )*/
    CALL_MAIN_ALGORETHM(PS_P_VAL);
    STATE_SYMBOL(WAITING());
}
STATE_DEFINE(WAITING)
{
    int i=0;
    Delay(50000);
    STATE_SYMBOL(READING());
}
```

2- MAIN ALGORITHM:



main_algorithm.h

```
#ifndef _MA_H_
#define _MA_H_

#include "state.h"
#include"stdio.h"

void (*MA_P)();

STATE_DEFINE(DETECTION);

#endif /*_MA_H_*/
```

main_algorithm.c

```
#include "main_algorithm.h"

int threshold=20;
int M_P_VAL=0;



STATE_DEFINE(DETECTION)
{
    CALL_ALARM();
}
void CALL_MAIN_ALGORETHM(int val)
{
    //printf("pressure value from s
    M_P_VAL = val;
    if(M_P_VAL<threshold)
    {}
    else
    {
        //printf("pressure value wi
        STATE_SYMBOL(DETECTION());
    }
}
```

3- ALARM MONITOR:

alarm_monitor.h

```h
#ifndef _ALARM_H_
#define _ALARM_H_
#include"state.h"
void (*A_M_P)();
STATE_DEFINE(ALARM_ON);
STATE_DEFINE(ALARM_OFF);
#endif /*_ALARM_H_*/

```

alarm_monitor.c

```c
#include"alarm_monitor.h"
void CALL_ALARM()
{
    //printf("main algorithm de
    STATE_SYMBOL(ALARM_ON());
}
STATE_DEFINE(ALARM_ON)
{
    Set_Alarm_actuator(0);
    Delay(500000);
    STATE_SYMBOL(ALARM_OFF());
}
STATE_DEFINE(ALARM_OFF)
{
    Set_Alarm_actuator(1);
}
```

4- DRIVER:

driver.h

```h
#ifndef _DRIVER_H_
#define _DRIVER_H_
#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 ();
#endif /*_DRIVER_H_*/
```

driver.c

```c
#include "driver.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;
}
```