Table of Contents:

1-introuction

This project is about how to detect high pressure in cabin and inform the crew of cabin with an alarm when the pressure exceeds 20 bars and the alarm duration equals 60 seconds.

2- system architecting /design sequence

2.1 case study (pressure controlling system)

-" **client**" expects you 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.

  - keeps track of the measured values.

- Pressure Controller: Assumptions

  The controller set up and shutdown procedures are not modeled

  The controller maintenance is not modeled

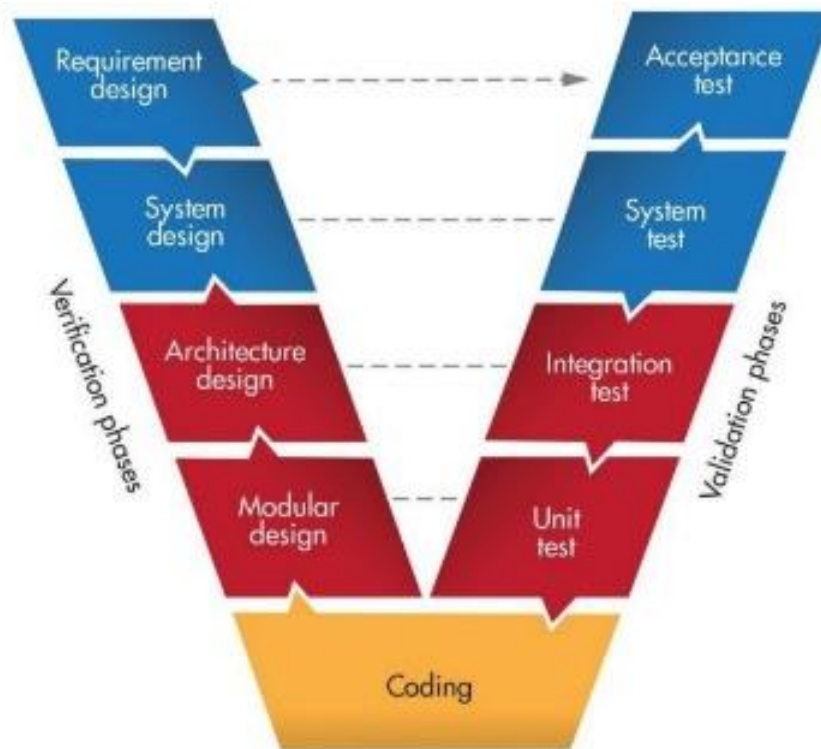 The pressure sensor never fails

  The alarm never fails

  The controller never faces power cut

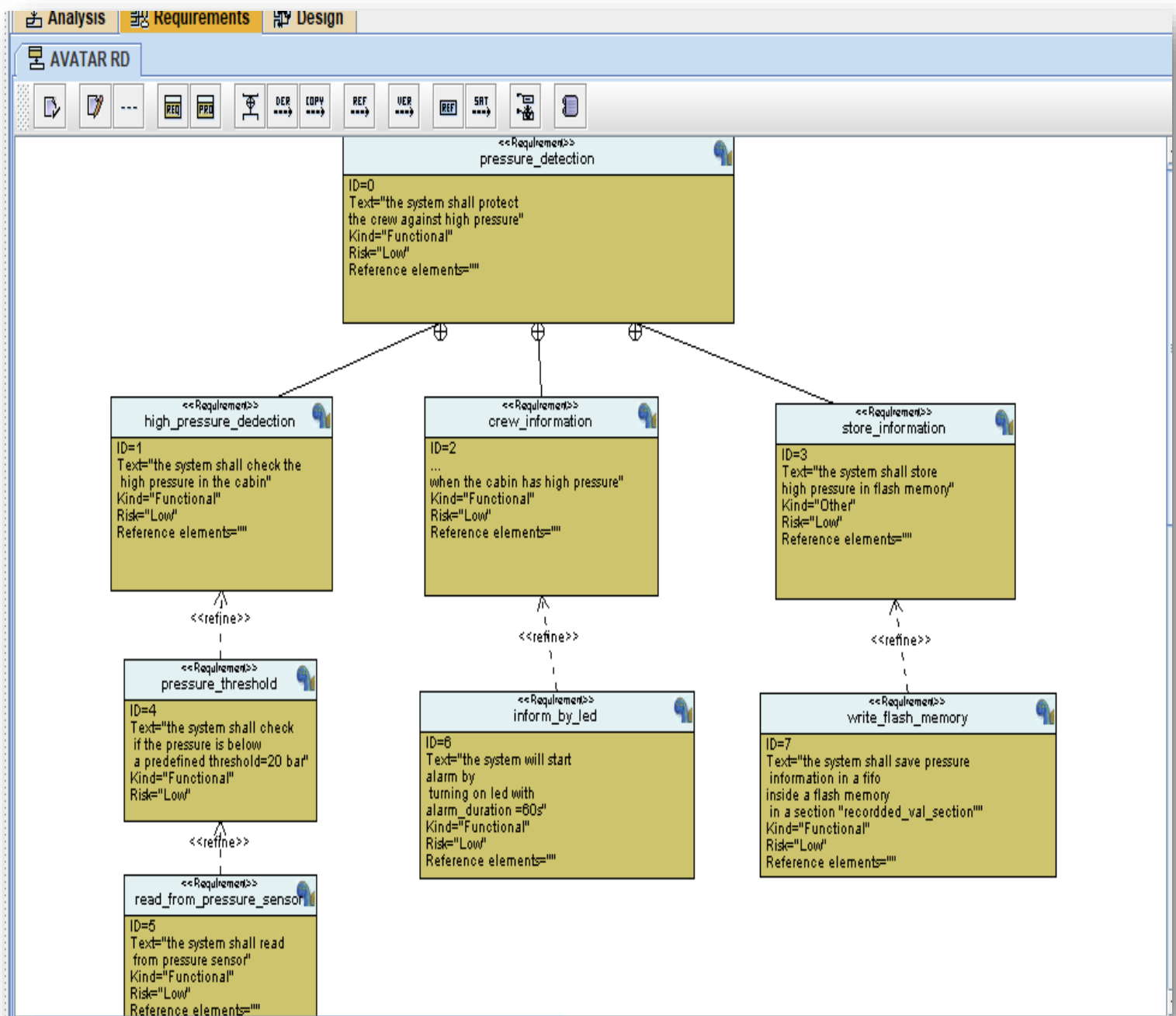- **Versioning•** The" keep track of measured value" option is not modeled in the first version of the design

2.2 method (SDLC &STLC)

starting with v-model for example
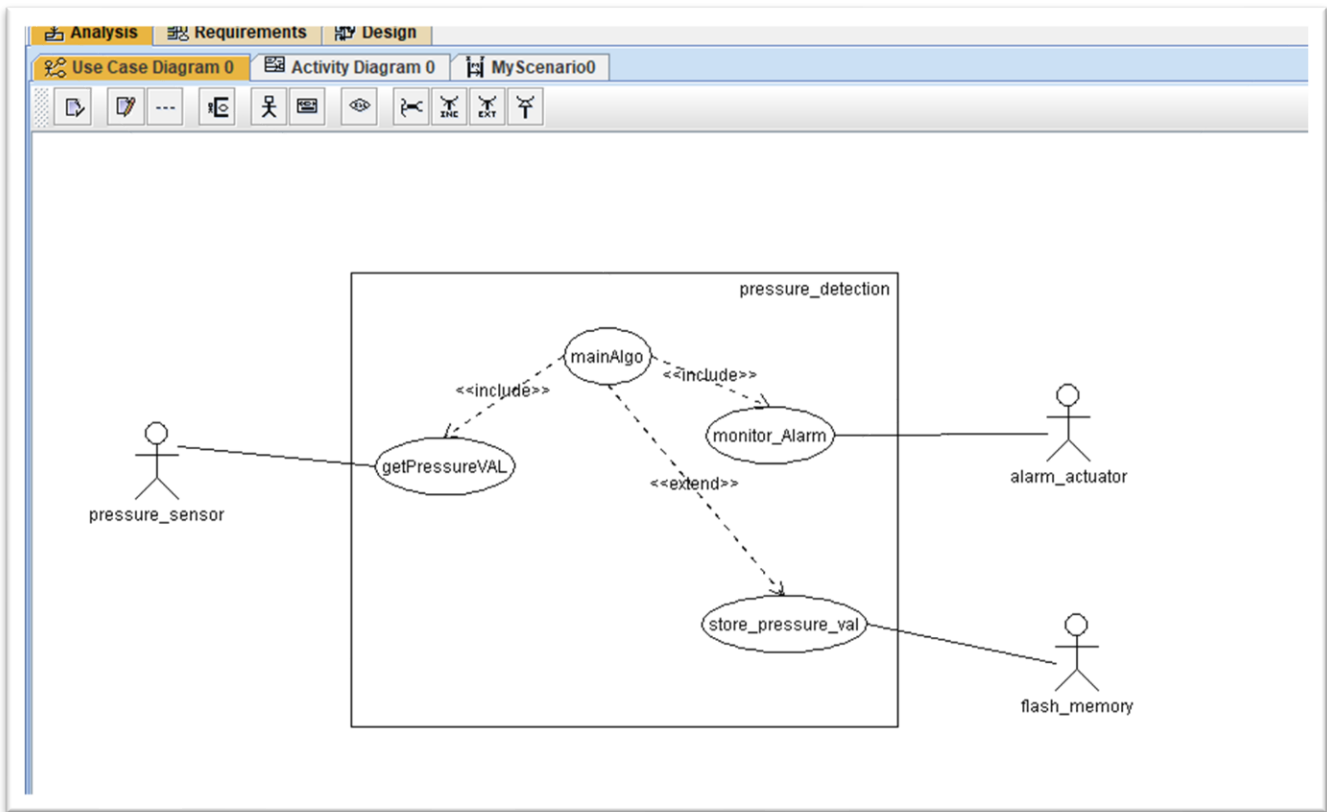
2.3 system requirement

2.4 space exploration/partitioning

How to chose MCs and number of MCs.

2.5 System analysis

2.5.1 use case diagram

2.5.2 Activity diagram
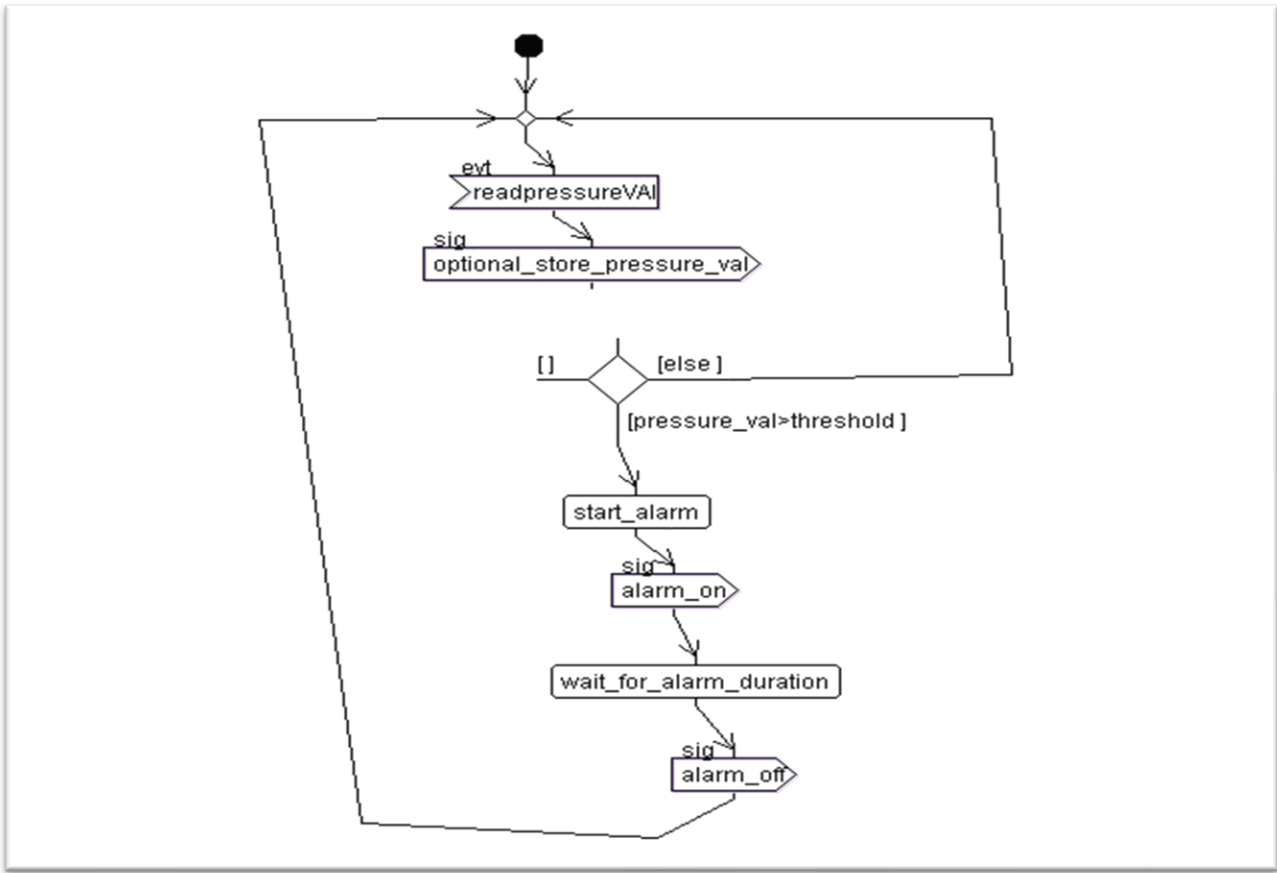
2.5.3 sequence diagram
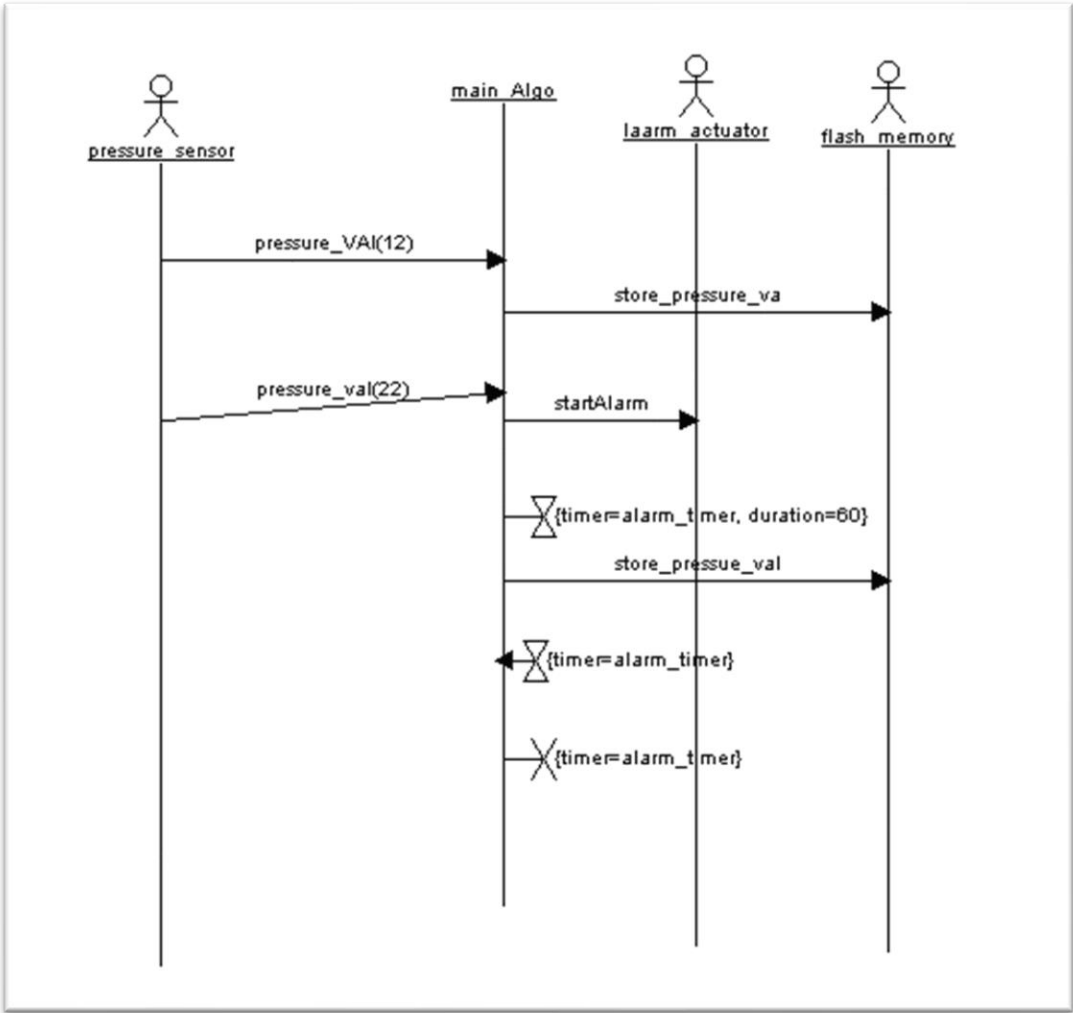
2.6 system design
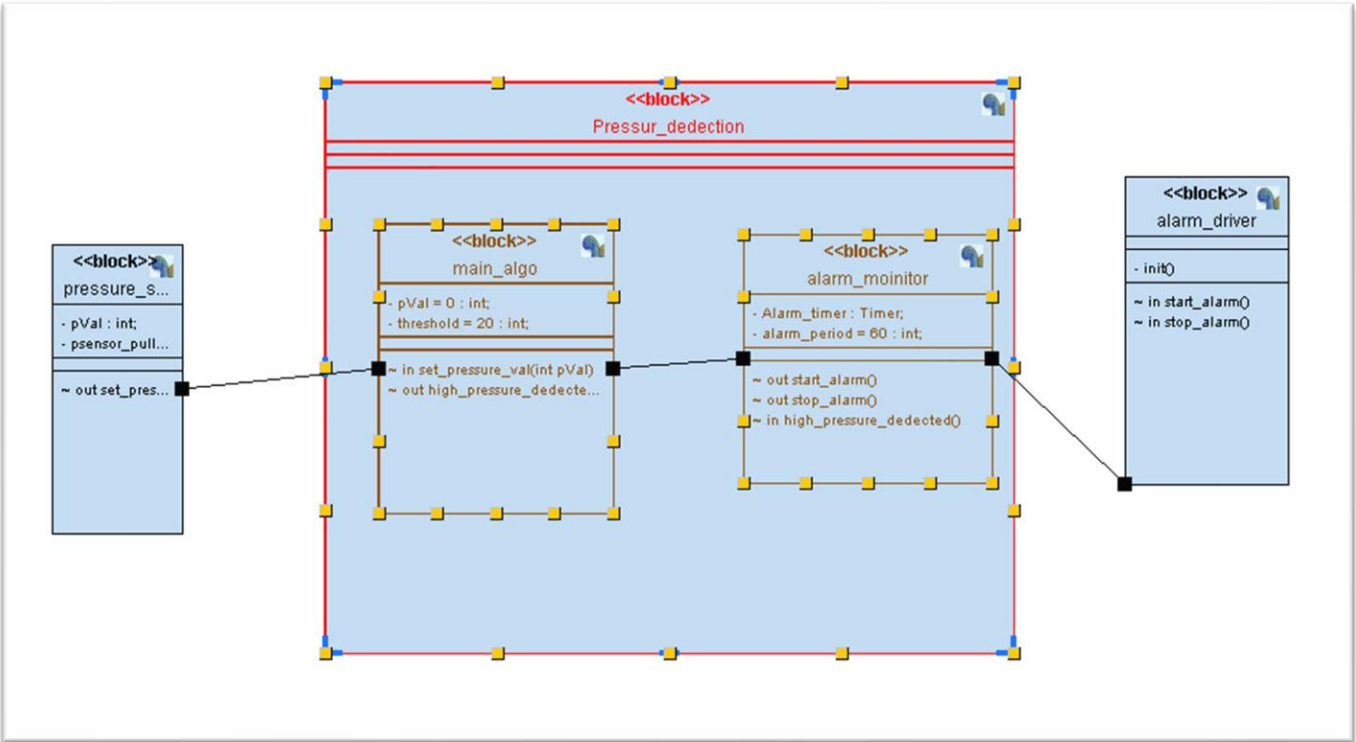
Pressure detection state diagram

init

reading

pVal = RANDOM0[15, 25]

set_pressure_val(pVal)

setTimer(psensor_pull_timer,60)

waiting

expire(psensor_pull_timer)

reset(psensor_pull_timer)
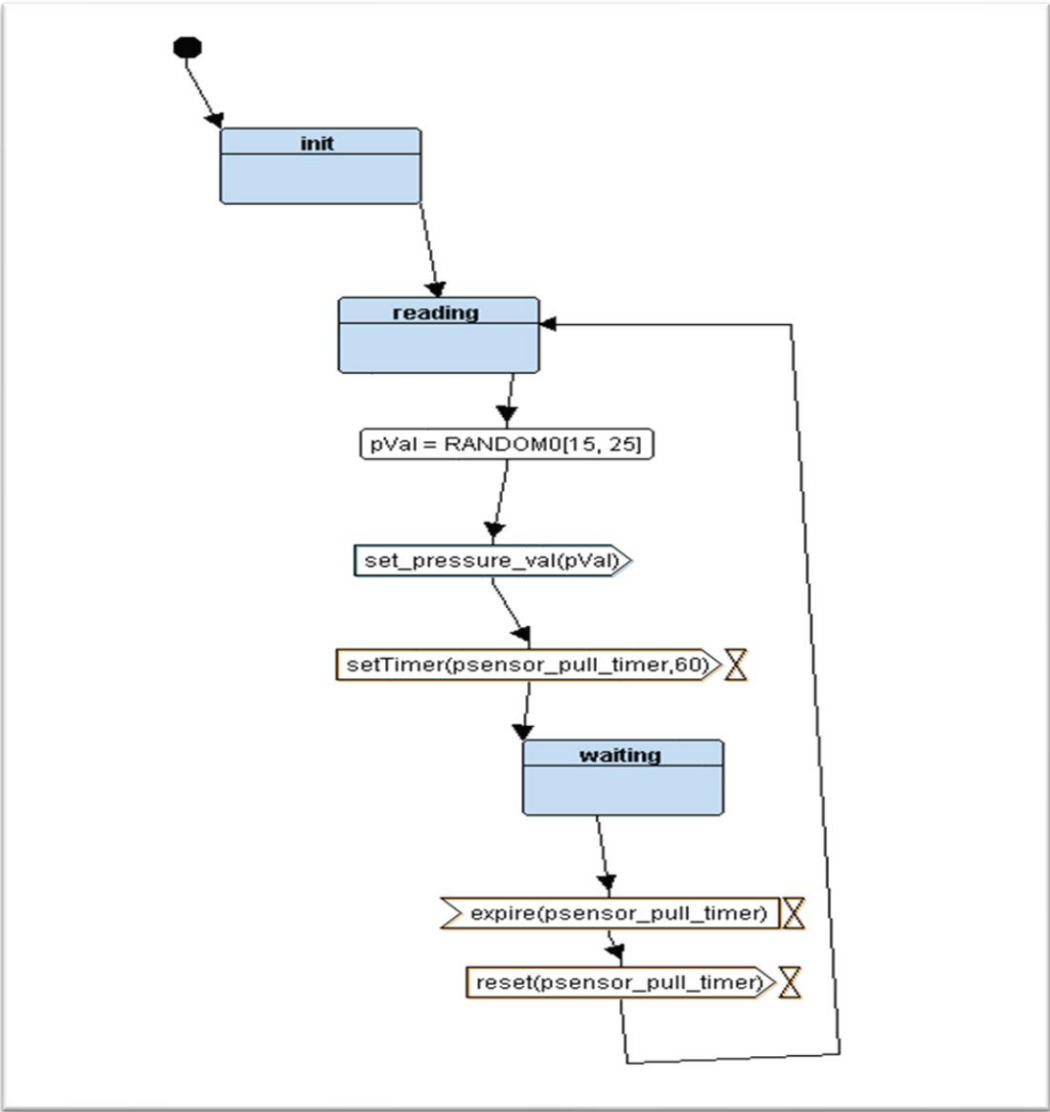
Main algorithm state diagram

set_pressure_val(pVal)

high_pressure_dedect

[pVal <= threshold]

[pVal > threshold]

set_pressure_val(pVal)

high_pressure_dedected()

set_pressure_val(pVal)
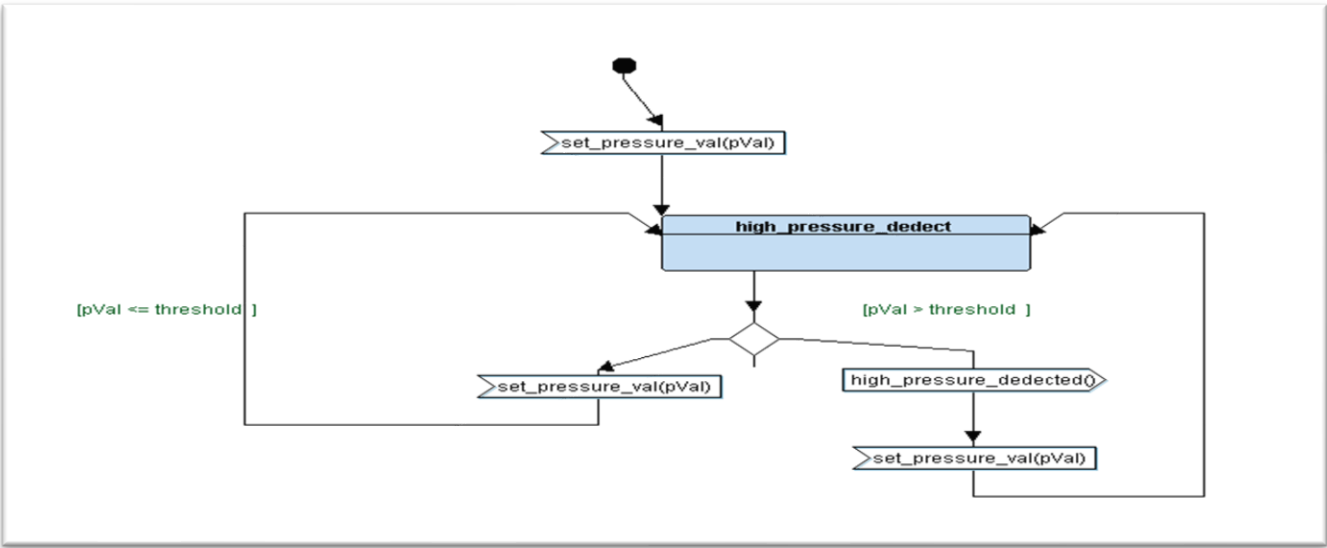
Alarm monitoring state diagram
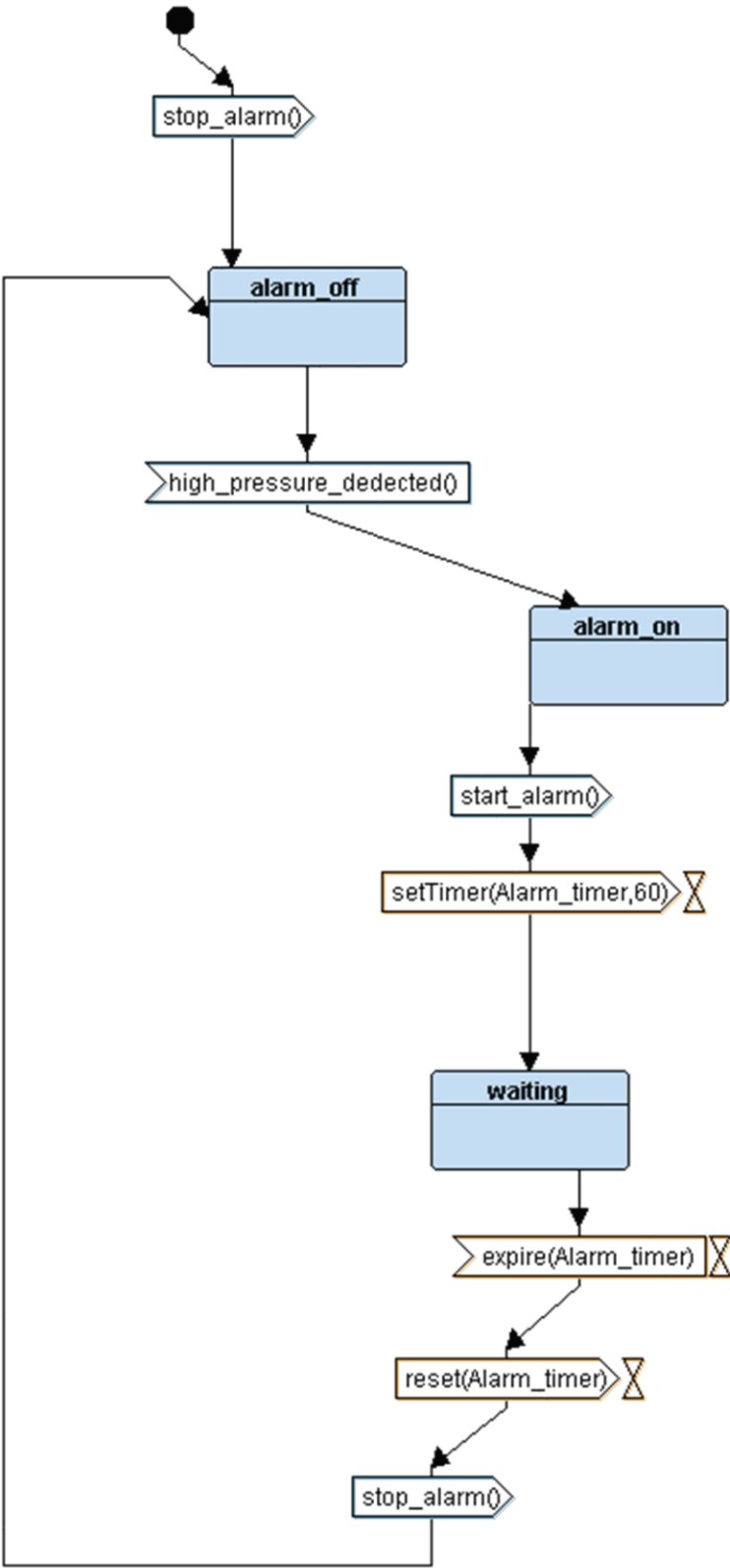
Alarm driver state diagram

2.7 system vervication

| alarm_driver | pressure_sensor_driver | alarm_moinitor | main_algo | Timer__psensor_pull_timer__pressure_sensor_driver | Timer__Alarm_timer__alarm_moinitor |
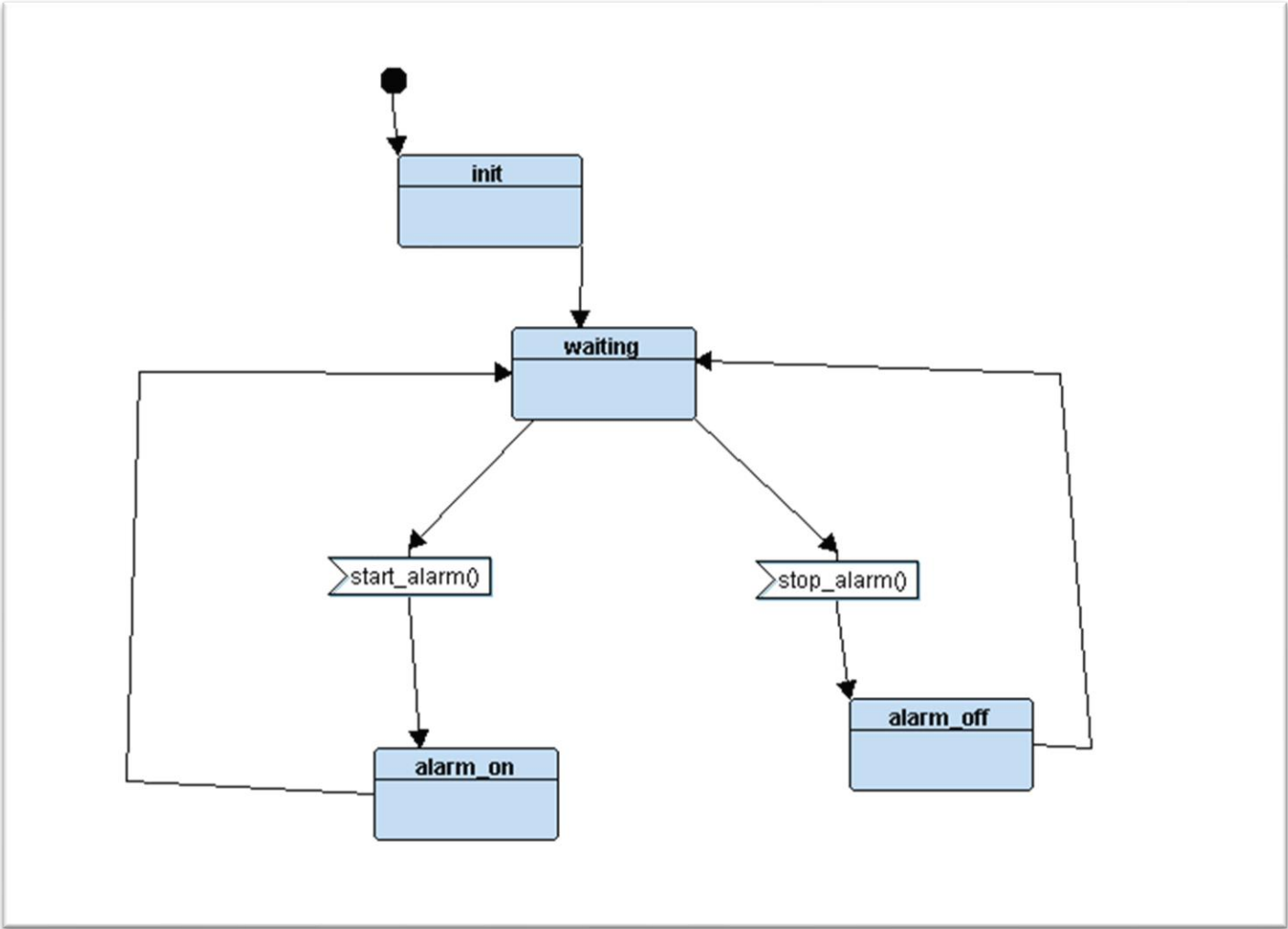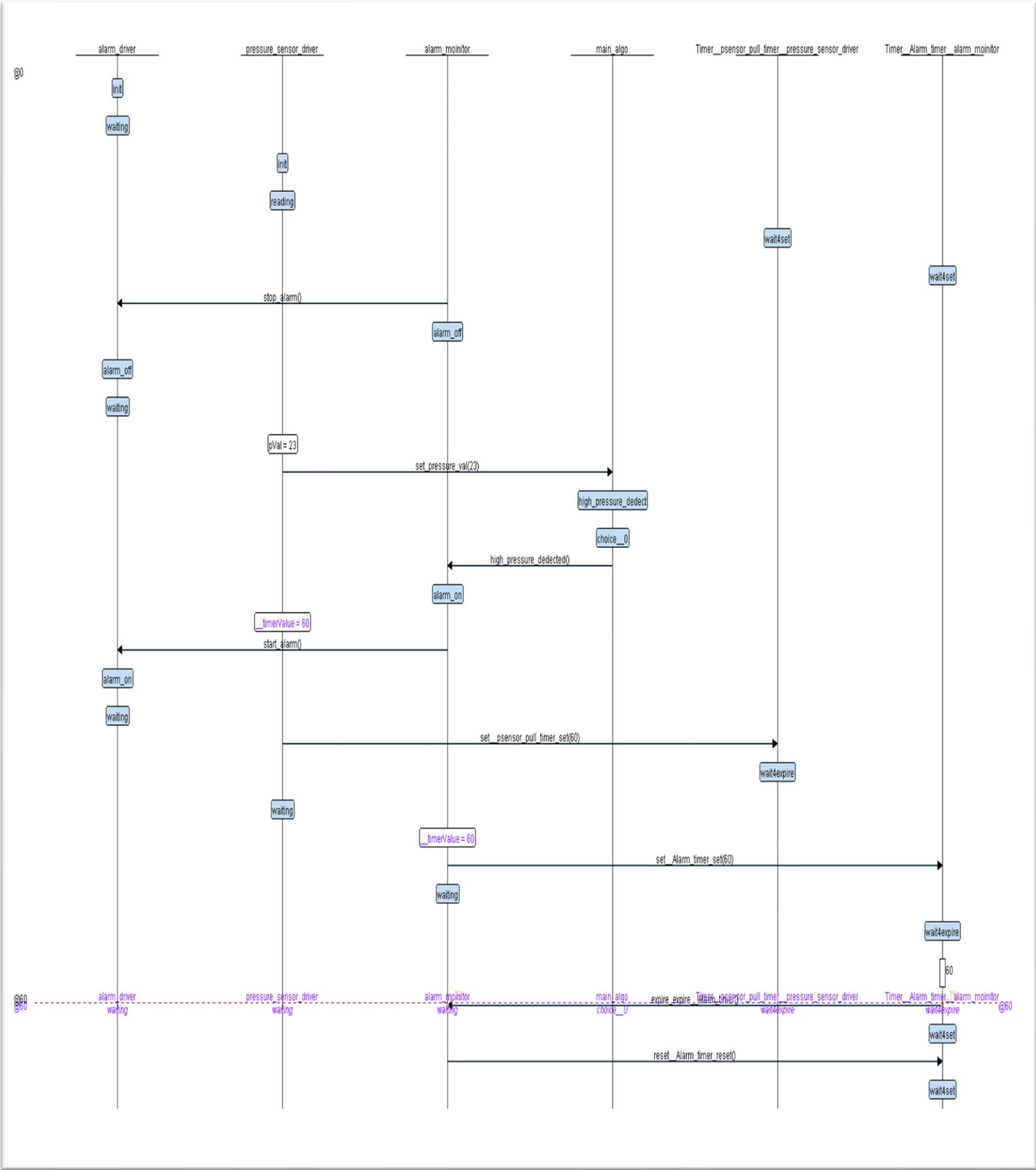
3- source code

3.0- make file

```
File   Edit   Format   View   Help
# eng <ahmed>
cc= arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -mthumb -gdwarf-2
INCS = -I .
LIBS =
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
As = $(wildcard *.s)
As_OBJ = $(As:.s=.o)
PROJECT_NAME=first_term_project1
all: $(PROJECT_NAME).bin
        @echo "---------all is done--------"
startup.o: startup.s
        $(cc)as.exe  $(CFLAGS) $(INCS) $< -o $@
%.o: %.c
        $(cc)gcc.exe -c $(INCS) $(CFLAGS) $< -o $@

$(PROJECT_NAME).elf: $(OBJ) $(As_OBJ)
        $(cc)ld.exe -T  linker_script.ld $(LIBS) $(OBJ) $(As_OBJ) -o $@ -Map=map_file
$(PROJECT_NAME).bin: $(PROJECT_NAME).elf
        $(cc)objcopy.exe -O binary $< $@
clean_all:
        rm *.o *.bin *.elf

clean:
        rm *.bin *.elf
```

3.1 – state.h

```c
1   /*
2    * state.h
3    *
4    *  Created on: Sep 10, 2022
5    *      Author: soft
6    */
7
8   #ifndef STATE_H_
9   #define STATE_H_
10
11  #include<stdio.h>
12  #include <stdlib.h>
13
14  //Automatic state function genrated
15  #define STATE_define(_statefunc_)  void ST_##_statefunc_()
16  #define STATE(_statefunc_)    ST_##_statefunc_
17
18  //state connections
19  int set_pressure_val(int pressure_Val);
20  int high_pressure_detected(int i);
21  void start_alarm();
22  void stop_alarm();
23
24  #endif /* STATE_H_ */
25
```

3.2- pressure_sensor.c/.h

```c
 4    *   Created on: Sep 10, 2022
 5    *       Author: soft
 6    */
 7   #include"pressure_sensor.h"
 8   #include "driver.h"
 9
10   //variables
11   int pressure_val=0;
12   /*int generate_random(int l, int r, int c){
13       int i;
14       for(i = 0; i < c; i++){
15           int rand_num = (rand() % (r-l+1)) + r;
16           return rand_num;
17       }
18   }*/
19   //global pointer
20   void(*pressure_state)();
21   //init
22   void pressure_init(){
23       //printf("pressure_init\n");
24       pressure_state=STATE(pressure_reading);
25   }
26
27   STATE_define(pressure_waiting){
28       //state name
29       pressure_sensor_state_id=pressure_waiting;
30       //state action
31       Delay(200000);
32       pressure_state=STATE(pressure_reading);
33       //check event
34       //printf("pressure_waiting\n");
35   }
36   STATE_define(pressure_reading){
37       //state name
38       pressure_sensor_state_id=pressure_reading;
39       //state action
40       pressure_val=getPressureVal();
41       //pressure_val=generate_random(15,25,1);
42       //check event
43       //printf("pressure_reading\n");
44       set_pressure_val(pressure_val);
45       pressure_state=STATE(pressure_waiting);
46
47   }
```

```c
/*
 * pressure_sensor.h
 *
 *  Created on: Sep 10, 2022
 *      Author: soft
 */

#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_

#include"state.h"

//states
enum {
    pressure_waiting,
    pressure_reading
}pressure_sensor_state_id;

//declare state function
STATE_define(pressure_waiting);
STATE_define(pressure_reading);
void pressure_init();

extern void(*pressure_state)();



#endif /* PRESSURE_SENSOR_H_ */
```

```c
/*
 * main_algo.c
 *
 *  Created on: Sep 10, 2022
 *      Author: soft
 */

#include"main_algo.h"

//variables
int pVAL=0;
int threshold=20;
//global pointer
void (*pmain_algo)()=STATE(idle);
int set_pressure_val(int pressure_Val);
//states
STATE_define(high_pressure_dedect){
    main_algo_state_id=high_pressure_dedect;
    start_alarm();
    //printf("high_pressure_dedect\n");
    high_pressure_detected(1);
    pmain_algo=STATE(idle);


}
STATE_define(idle){
    set_alarm_actuator(0);
    stop_alarm();
    high_pressure_detected(0);
    main_algo_state_id=idle;
    //printf("idle\n");

}

int set_pressure_val(int pressure_Val){
    if(pressure_Val>threshold){

            pmain_algo=STATE(high_pressure_dedect);
        }
        else if(pVAL<=threshold)
        {
            pmain_algo=STATE(idle);
        }

}
```

```c
/*
 * main_algo.h
 *
 *  Created on: Sep 11, 2022
 *      Author: soft
 */

#ifndef MAIN_ALGO_H_
#define MAIN_ALGO_H_

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

//states
enum {
    idle,
    high_pressure_dedect
}main_algo_state_id;

// state function
STATE_define(high_pressure_dedect);
STATE_define(idle);

//global pointer
extern void (*pmain_algo)();


#endif /* MAIN_ALGO_H_ */
```

3.4- alarm_moinitor.c/.h

```c
/*
 * alarm_moinitor.h
 *
 *  Created on: Sep 10, 2022
 *      Author: soft
 */

#ifndef ALARM_MOINITOR_H_
#define ALARM_MOINITOR_H_

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

//states
enum {
    alarm_moinitor_off,
    alarm_moinitor_on,
    alarm_moinitor_waiting
}alarm_monitor_state_id;

//functions
STATE_define(alarm_off);
STATE_define(alarm_on);
STATE_define(alarm_waiting);

//pointer
extern void (*pMoinitor_alarm)();

#endif /* ALARM_MOINITOR_H_ */
```

```c
#include "alarm_moinitor.h"

//variables
//int alarm_period=60;
//int timer;

//states pointer
void (*pMoinitor_alarm)()=STATE(alarm_off);

//states function
STATE_define(alarm_off){
    alarm_monitor_state_id=alarm_moinitor_off;
    stop_alarm();
    //printf("alarm_off\n");

}
STATE_define(alarm_on){
    alarm_monitor_state_id=alarm_moinitor_on;
    start_alarm();
    //printf("alarm_on\n");
    pMoinitor_alarm=STATE(alarm_waiting);
}
STATE_define(alarm_waiting){
    alarm_monitor_state_id=alarm_moinitor_waiting;
    Delay(200000);
    //printf("alarm_waiting\n");
    pMoinitor_alarm=STATE(alarm_off);

}
int high_pressure_detected(int i){

    //((i==1)?pMoinitor_alarm=STATE(alarm_on):pMoinitor_alarm=STATE(alarm_off));
    if(i==1){
        pMoinitor_alarm=STATE(alarm_on);
    }
    else
    {
        pMoinitor_alarm=STATE(alarm_off);
    }
}
```

3.5 – alarm_driver.c/.h

```c
/*
 * alarm_driver.h
 *
 *  Created on: Sep 10, 2022
 *      Author: soft
 */

#ifndef ALARM_DRIVER_H_
#define ALARM_DRIVER_H_

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

enum {
    alarm_driver_waiting,
    alarm_driver_on,
    alarm_driver_off
}alarm_driver_state_id;

STATE_define(alarm_driver_waiting);
STATE_define(alarm_driver_on);
STATE_define(alarm_driver_off);

extern int (*palarm_driver)();
void alarm_init();
#endif /* ALARM_DRIVER_H_ */
```

```c
#include "alarm_driver.h"

int (*palarm_driver)();
void alarm_init(){
    //printf("alarm_init\n");
    palarm_driver=STATE(alarm_driver_waiting);
}
void start_alarm(){
    //printf("start_alarm\n");
    set_alarm_actuator(1);
}
void stop_alarm(){
    set_alarm_actuator(0);
    //printf("stop_alarm\n");
}
STATE_define(alarm_driver_waiting){
    Delay(200000);
    //printf("alarm_driver_waiting\n");
}
STATE_define(alarm_driver_on){
    start_alarm();
    palarm_driver=STATE(alarm_driver_waiting);
}
STATE_define(alarm_driver_off){
    stop_alarm();
    palarm_driver=STATE(alarm_driver_waiting);
}
```

3.6 – driver.c/.h

```c
/*
 * driver.c
 *
 *  Created on: Sep 10, 2022
 *      Author: soft
 */

#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)
    {
        CLEAR_BIT(GPIOA_ODR,13);
    }
}
void GPIO_INITILAIZATION(){
    SET_BIT(APB2ENR,2);
    GPIOA_CRL &= 0xff0fffff;
    GPIOA_CRL |= 0x00000000;
    GPIOA_CRH &= 0xff0fffff;
    GPIOA_CRH |= 0x22222222;
}
```

```c
1   /*
2    * driver.h
3    *
4    *  Created on: Sep 10, 2022
5    *      Author: soft
6    */
7
8   #ifndef DRIVER_H_
9   #define DRIVER_H_
10
11  #include <stdint.h>
12  #include <stdio.h>
13
14  #define SET_BIT(ADDRESS,BIT)    ADDRESS|=(1<<BIT)
15  #define CLEAR_BIT(ADDRESS,BIT)  ADDRESS &= ~(1<<BIT)
16  #define TOGGLE_BIT(ADDRESS,BIT) ADDRESS ^= (1<<BIT)
17  #define READ_BIT(ADDRESS,BIT)   ((ADDRESS) & (1<<BIT))
18
19  #define GPIO_PORTA  0X40010800
20  #define BASE_RCC    0X40021000
21
22  #define APB2ENR *(volatile uint32_t*)(BASE_RCC + 0X18)
23
24  #define GPIOA_CRL *(volatile uint32_t*)(GPIO_PORTA + 0X00)
25  #define GPIOA_CRH  *(volatile uint32_t*)(GPIO_PORTA + 0X04)
26  #define GPIOA_IDR  *(volatile uint32_t*)(GPIO_PORTA + 0X08)
27  #define GPIOA_ODR  *(volatile uint32_t*)(GPIO_PORTA + 0X0C)
28  //APIS
29  void Delay(int ncount);
30  int getPressureVal();
31  void set_alarm_actuator(int i);
32  void GPIO_INITILAIZATION();
33
34  #endif /* DRIVER_H_ */
35
```

3.7-main.c

```c
1   /*
2    * main.c
3    *
4    *  Created on: Sep 11, 2022
5    *      Author: soft
6    */
7
8
9   #include "alarm_driver.h"
10  #include "alarm_moinitor.h"
11  #include "main_algo.h"
12  #include "pressure_sensor.h"
13
14  void setup(){
15      GPIO_INITILAIZATION();
16      pressure_init();
17      alarm_init();
18  }
19
20
21
22
23  int main(){
24      setup();
25      while(1){
26          pressure_state();
27          pmain_algo();
28          pMoinitor_alarm();
29          palarm_driver();
30          Delay(10000);
31      }
32
33
34
35
36
37      return 0;
38  }
39
```

3.8 – starup code with assembly

```asm
//startup.s
//Eng.Ahmed Mohsen

.section .vectors

.word    0x20001000       //stack top address
.word    _reset           //1 reset
.word    vector_handeler //2 NMI
.word    vector_handeler //3 HARD FAULT
.word    vector_handeler //4 MM FAULT
.word    vector_handeler //5 BUS FAULT
.word    vector_handeler //6 USAGE FAULT
.word    vector_handeler //7 RESERVED
.word    vector_handeler //8 RESERVED
.word    vector_handeler //9 RESERVED
.word    vector_handeler //10 RESERVED
.word    vector_handeler //11 SV CALL
.word    vector_handeler //12 DEBUG RESERVED
.word    vector_handeler //13 RESERVED
.word    vector_handeler //14 PENDSV
.word    vector_handeler //15 SYS TICK
.word    vector_handeler //16 IRQ0
.word    vector_handeler //17 IRQ1
.word    vector_handeler //18 IRG2
.word    vector_handeler //19 ...
         // on to IRQ67

.section  .text
_reset:
    bl main
    b .

    .thumb_func    //enable 16 and 32 bit
vector_handeler:
    b  _reset
```

3.9 -linker script

```ld
MEMORY
{
    flash(RX) : ORIGIN = 0X08000000, LENGTH = 128k
    sram(RWX) : ORIGIN = 0X20000000, LENGTH = 20k
}

SECTIONS
{
    .text :
    {
        *(.vectors*)
        *(.text*)
        *(.rodata)
        _E_text = .;
    } > flash
    .data :
    {
        _S_data = .;
        *(.data)
        _E_data = .;
    }> sram AT> flash
    .bss :
    {
        _S_bss = .;
        *(.bss*)
        . = ALIGN(4);
        _E_bss = .;

    } > sram

}
```

4- Symbols

4.1- pressure_sensor.o

```
         U Delay
         U getPressureVal
00000000 T pressure_init
00000001 C pressure_sensor_state_id
00000004 C pressure_state
00000000 B pressure_val
         U set_pressure_val
00000050 T ST_pressure_reading
0000001c T ST_pressure_waiting
```

4.2 – main_algo.o

```
         U high_pressure_detected
00000001 C main_algo_state_id
00000004 D pmain_algo
00000000 B pVAL
         U set_alarm_actuator
0000005c T set_pressure_val
00000000 T ST_high_pressure_dedect
00000034 T ST_idle
         U start_alarm
         U stop_alarm
00000000 D threshold
```

4.3- alarm_moinitor.o

```
00000001 C alarm_monitor_state_id
         U Delay
00000078 T high_pressure_detected
00000000 D pMoinitor_alarm
00000000 T ST_alarm_off
00000018 T ST_alarm_on
00000044 T ST_alarm_waiting
         U start_alarm
         U stop_alarm
```

4.4 – alarm_driver.o

```
$ arm-none-eabi-nm.exe alarm_driver.o
00000001 C alarm_driver_state_id
00000000 T alarm_init
         U Delay
00000004 C palarm_driver
         U set_alarm_actuator
0000006c T ST_alarm_driver_off
00000050 T ST_alarm_driver_on
0000003c T ST_alarm_driver_waiting
0000001c T start_alarm
0000002c T stop_alarm
```

4.5 – driver.o

```
00000000 T Delay
00000024 T getPressureVal
0000008c T GPIO_INITILAIZATION
0000003c T set_alarm_actuator
```

4.6-main.o

```
00000001 C alarm_driver_state_id
         U alarm_init
00000001 C alarm_monitor_state_id
         U Delay
         U GPIO_INITILAIZATION
00000014 T main
00000001 C main_algo_state_id
         U palarm_driver
         U pmain_algo
         U pMoinitor_alarm
         U pressure_init
00000001 C pressure_sensor_state_i
         U pressure_state
00000000 T setup
```

4.7 – starup .o

```
00000000 t _reset
         U main
00000006 t vector_handeler
```

4.8- elf image symbols

```
0800044c t _reset
20000008 B alarm_driver_state_id
08000050 T alarm_init
20000010 B alarm_monitor_state_id
08000190 T Delay
080001b4 T getPressureVal
0800021c T GPIO_INITILAIZATION
08000150 T high_pressure_detected
080002b0 T main
20000011 B main_algo_state_id
2000000c B palarm_driver
0800045c D pmain_algo
08000454 D pMoinitor_alarm
080003b4 T pressure_init
20000012 B pressure_sensor_state_id
20000014 B pressure_state
20000004 B pressure_val
20000000 B pVAL
080001cc T set_alarm_actuator
08000350 T set_pressure_val
0800029c T setup
080000bc T ST_alarm_driver_off
080000a0 T ST_alarm_driver_on
0800008c T ST_alarm_driver_waiting
080000d8 T ST_alarm_off
080000f0 T ST_alarm_on
0800011c T ST_alarm_waiting
080002f4 T ST_high_pressure_dedect
08000328 T ST_idle
08000404 T ST_pressure_reading
080003d0 T ST_pressure_waiting
0800006c T start_alarm
0800007c T stop_alarm
08000458 D threshold
08000452 t vector_handeler
```

5- Sections Headers

5.1- pressure sensor

```
pressure_sensor.o:      file format elf32-littlearm

Sections:
Idx Name            Size      VMA       LMA       File off  Algn
  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            00000004  00000000  00000000  000000cc  2**2
                    ALLOC
  3 .debug_info     00000111  00000000  00000000  000000cc  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev   00000092  00000000  00000000  000001dd  2**0
                    CONTENTS, READONLY, DEBUGGING
  5 .debug_loc      00000084  00000000  00000000  0000026f  2**0
                    CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges  00000020  00000000  00000000  000002f3  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line     00000063  00000000  00000000  00000313  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str      00000188  00000000  00000000  00000376  2**0
                    CONTENTS, READONLY, DEBUGGING
  9 .comment        00000012  00000000  00000000  000004fe  2**0
                    CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00000510  2**0
                    CONTENTS, READONLY
 11 .debug_frame    00000060  00000000  00000000  00000544  2**2
                    CONTENTS, RELOC, READONLY, DEBUGGING
```

5.2 – main algo

```
main_algo.o:      file format elf32-littlearm

Sections:
Idx Name            Size      VMA       LMA       File off  Algn
  0 .text           000000c0  00000000  00000000  00000034  2**2
                    CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data           00000008  00000000  00000000  000000f4  2**2
                    CONTENTS, ALLOC, LOAD, RELOC, DATA
  2 .bss            00000004  00000000  00000000  000000fc  2**2
                    ALLOC
  3 .debug_info     0000013b  00000000  00000000  000000fc  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev   000000a7  00000000  00000000  00000237  2**0
                    CONTENTS, READONLY, DEBUGGING
  5 .debug_loc      00000090  00000000  00000000  000002de  2**0
                    CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges  00000020  00000000  00000000  0000036e  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line     0000005b  00000000  00000000  0000038e  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str      0000017a  00000000  00000000  000003e9  2**0
                    CONTENTS, READONLY, DEBUGGING
  9 .comment        00000012  00000000  00000000  00000563  2**0
                    CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00000575  2**0
                    CONTENTS, READONLY
 11 .debug_frame    00000064  00000000  00000000  000005a8  2**2
                    CONTENTS, RELOC, READONLY, DEBUGGING
```

5.3- alarm_moinitor

```
alarm_moinitor.o:     file format elf32-littlearm

Sections:
Idx Name            Size      VMA       LMA       File off  Algn
  0 .text           000000b8  00000000  00000000  00000034  2**2
                    CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data           00000004  00000000  00000000  000000ec  2**2
                    CONTENTS, ALLOC, LOAD, RELOC, DATA
  2 .bss            00000000  00000000  00000000  000000f0  2**0
                    ALLOC
  3 .debug_info     00000130  00000000  00000000  000000f0  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev   000000a7  00000000  00000000  00000220  2**0
                    CONTENTS, READONLY, DEBUGGING
  5 .debug_loc      000000bc  00000000  00000000  000002c7  2**0
                    CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges  00000020  00000000  00000000  00000383  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line     00000066  00000000  00000000  000003a3  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str      0000019e  00000000  00000000  00000409  2**0
                    CONTENTS, READONLY, DEBUGGING
  9 .comment        00000012  00000000  00000000  000005a7  2**0
                    CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000005b9  2**0
                    CONTENTS, READONLY
 11 .debug_frame    00000080  00000000  00000000  000005ec  2**2
                    CONTENTS, RELOC, READONLY, DEBUGGING
```

5.4 – alarm driver

```
alarm_driver.o:     file format elf32-littlearm

Sections:
Idx Name            Size      VMA       LMA       File off  Algn
  0 .text           00000088  00000000  00000000  00000034  2**2
                    CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data           00000000  00000000  00000000  000000bc  2**0
                    CONTENTS, ALLOC, LOAD, DATA
  2 .bss            00000000  00000000  00000000  000000bc  2**0
                    ALLOC
  3 .debug_info     00000148  00000000  00000000  000000bc  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev   00000094  00000000  00000000  00000204  2**0
                    CONTENTS, READONLY, DEBUGGING
  5 .debug_loc      00000108  00000000  00000000  00000298  2**0
                    CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges  00000020  00000000  00000000  000003a0  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line     00000063  00000000  00000000  000003c0  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str      000001b3  00000000  00000000  00000423  2**0
                    CONTENTS, READONLY, DEBUGGING
  9 .comment        00000012  00000000  00000000  000005d6  2**0
                    CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000005e8  2**0
                    CONTENTS, READONLY
 11 .debug_frame    000000b4  00000000  00000000  0000061c  2**2
                    CONTENTS, RELOC, READONLY, DEBUGGING
```

5.5 – driver

```
driver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000010c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000140  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000140  2**0
                  ALLOC
  3 .debug_info   00000103  00000000  00000000  00000140  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 0000009d  00000000  00000000  00000243  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000000c8  00000000  00000000  000002e0  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020 00000000  00000000  000003a8  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000009e  00000000  00000000  000003c8  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000013e  00000000  00000000  00000466  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  000005a4  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033 00000000 00000000  000005b6  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000078  00000000  00000000  000005ec  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

5.6- main

```
main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000058  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000008c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000008c  2**0
                  ALLOC
  3 .debug_info   000001a2  00000000  00000000  0000008c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000ae  00000000  00000000  0000022e  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000058  00000000  00000000  000002dc  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020 00000000  00000000  00000334  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000000a4  00000000  00000000  00000354  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000023a  00000000  00000000  000003f8  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  00000632  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033 00000000 00000000  00000644  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000048  00000000  00000000  00000678  2**2
```

5.7 – startup

```
Sections:
Idx Name            Size      VMA       LMA       File off  Algn
  0 .text           00000008  00000000  00000000  00000034  2**1
                    CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data           00000000  00000000  00000000  0000003c  2**0
                    CONTENTS, ALLOC, LOAD, DATA
  2 .bss            00000000  00000000  00000000  0000003c  2**0
                    ALLOC
  3 .vectors        00000050  00000000  00000000  0000003c  2**0
                    CONTENTS, RELOC, READONLY
  4 .ARM.attributes 00000021  00000000  00000000  0000008c  2**0
                    CONTENTS, READONLY
  5 .debug_line     0000003b  00000000  00000000  000000ad  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  6 .debug_info     0000007d  00000000  00000000  000000e8  2**0
                    CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_abbrev   00000014  00000000  00000000  00000165  2**0
                    CONTENTS, READONLY, DEBUGGING
  8 .debug_aranges  00000020  00000000  00000000  00000180  2**3
                    CONTENTS, RELOC, READONLY, DEBUGGING
```

5.8- elf image symbols 6- run application on protus

```
first_term_project1.elf:      file format elf32-littlearm

Sections:
Idx Name            Size      VMA       LMA       File off  Algn
  0 .text           00000454  08000000  08000000  00008000  2**2
                    CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data           0000000c  08000454  08000454  00008454  2**2
                    CONTENTS, ALLOC, LOAD, DATA
  2 .bss            00000018  20000000  20000000  00010000  2**2
                    ALLOC
  3 .debug_info     000007e6  00000000  00000000  00008460  2**0
                    CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev   000003d3  00000000  00000000  00008c46  2**0
                    CONTENTS, READONLY, DEBUGGING
  5 .debug_loc      000003f8  00000000  00000000  00009019  2**0
                    CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges  000000e0  00000000  00000000  00009418  2**3
                    CONTENTS, READONLY, DEBUGGING
  7 .debug_line     00000304  00000000  00000000  000094f8  2**0
                    CONTENTS, READONLY, DEBUGGING
  8 .debug_str      0000033e  00000000  00000000  000097fc  2**0
                    CONTENTS, READONLY, DEBUGGING
  9 .comment        00000011  00000000  00000000  00009b3a  2**0
                    CONTENTS, READONLY
 10 .ARM.attributes 00000031  00000000  00000000  00009b4b  2**0
                    CONTENTS, READONLY
 11 .debug_frame    000002b8  00000000  00000000  00009b7c  2**2
                    CONTENTS, READONLY, DEBUGGING
```

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: Ahmed Abd El-mohsen

Pressure Sensor

R1 10k   R2 10k   R3 10k   R4 10k   R5 10k   R6 10k   R7 10k   R8 10k

Bit 0

Bit 7

R10
100

ALARM   D2   LED-YELLOW

U1
PA0-WKUP   NRST
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15   PC13_RTC
PC14-OSC32_IN
PC15-OSC32_OUT
PB0   OSCIN_PD0
PB1   OSCOUT_PD1
PB2
PB3
PB4
PB5
PB6
PB7
PB8
PB9
PB10
PB11   VBAT
PB12
PB13
PB14
PB15   BOOT0
STM32F103C6
VDDA=VDD
VSSA=VSS