Mastering Embedded System Online Diploma

www.learn-in-depth.com

First Term (Final Project1)

Report for “High_Pressure_Detection” project

Name: Ahmed Basem Mohamed

Mail: Basem010104@gmail.com

Githup repo:

https://github.com/Basem0/Master-Embedded-Systems/

My profile:

https://www.learn-in-depth-store.com/certificate/basem010104%40gmail.com

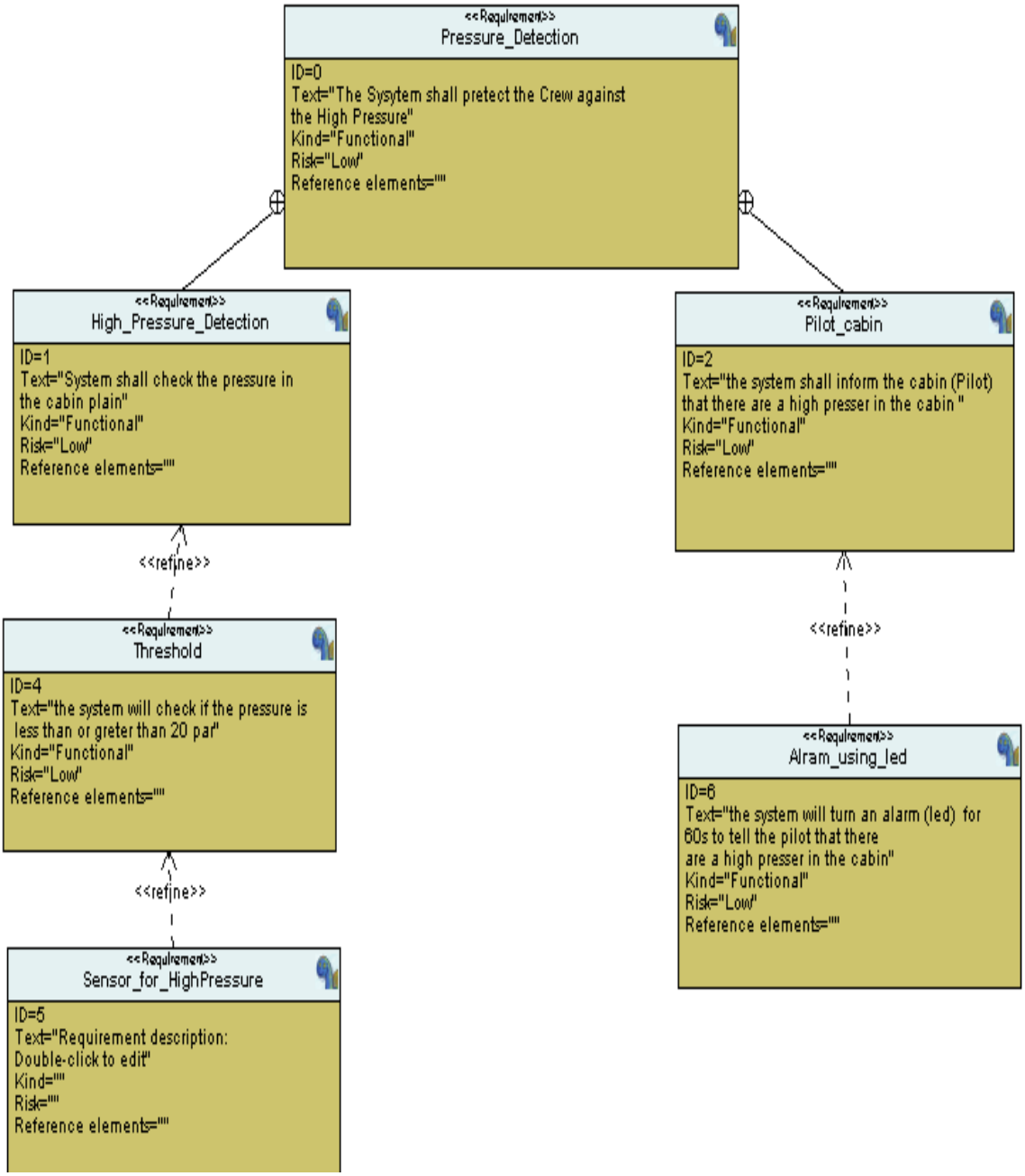
Case Study

The system should be in a plain cabin, if the pressure in the cabin is higher we will put a sensor -to sense the high pressure- and connected with an Alarm to tell the crew in the cabin that there is a high pressure to be safe

The Sensor should make Alarm if the pressure is greater than 20 bar.

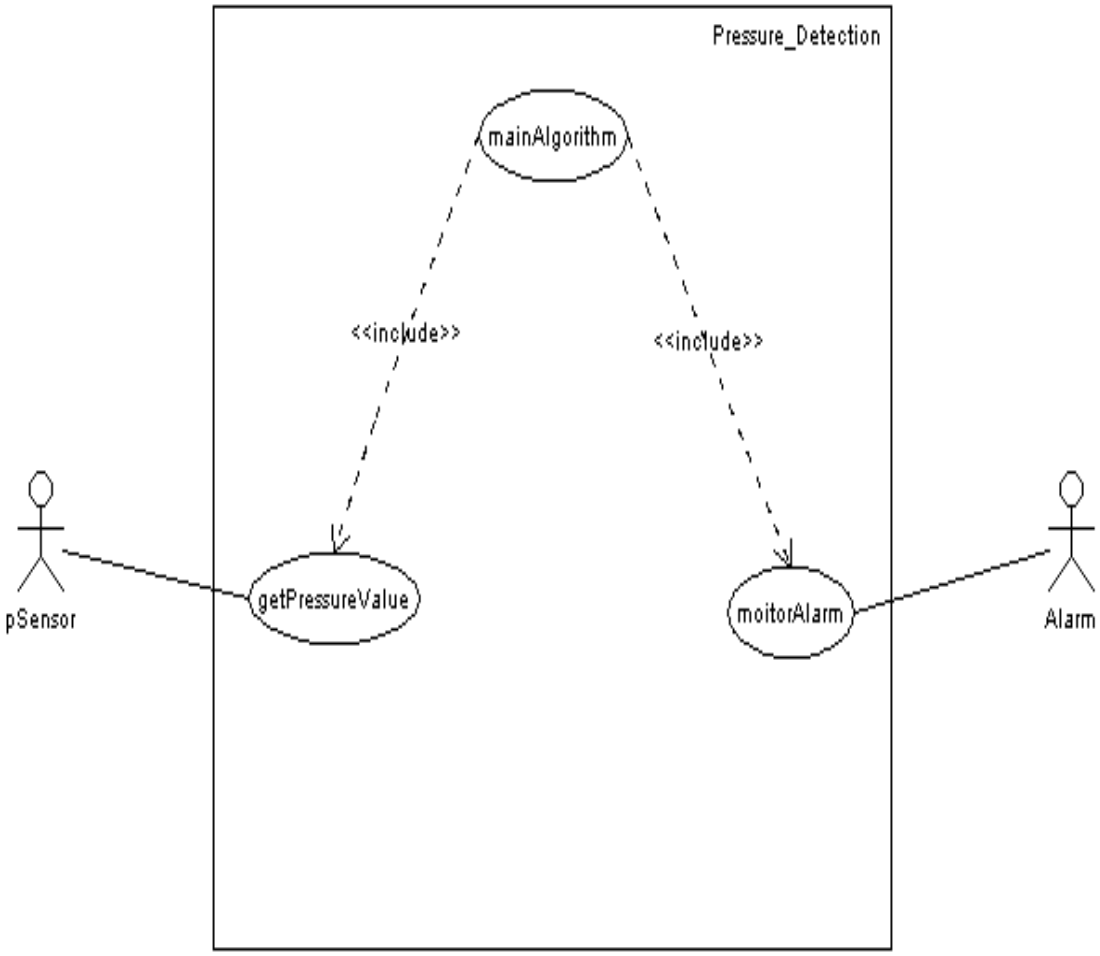
Then the Alarm -The led- will turn on 60s and the Alarm still turn on if the pressure is greater than 20bar to tell the crew or the Pilot that there are a danger on the plain.

Requirements Diagram

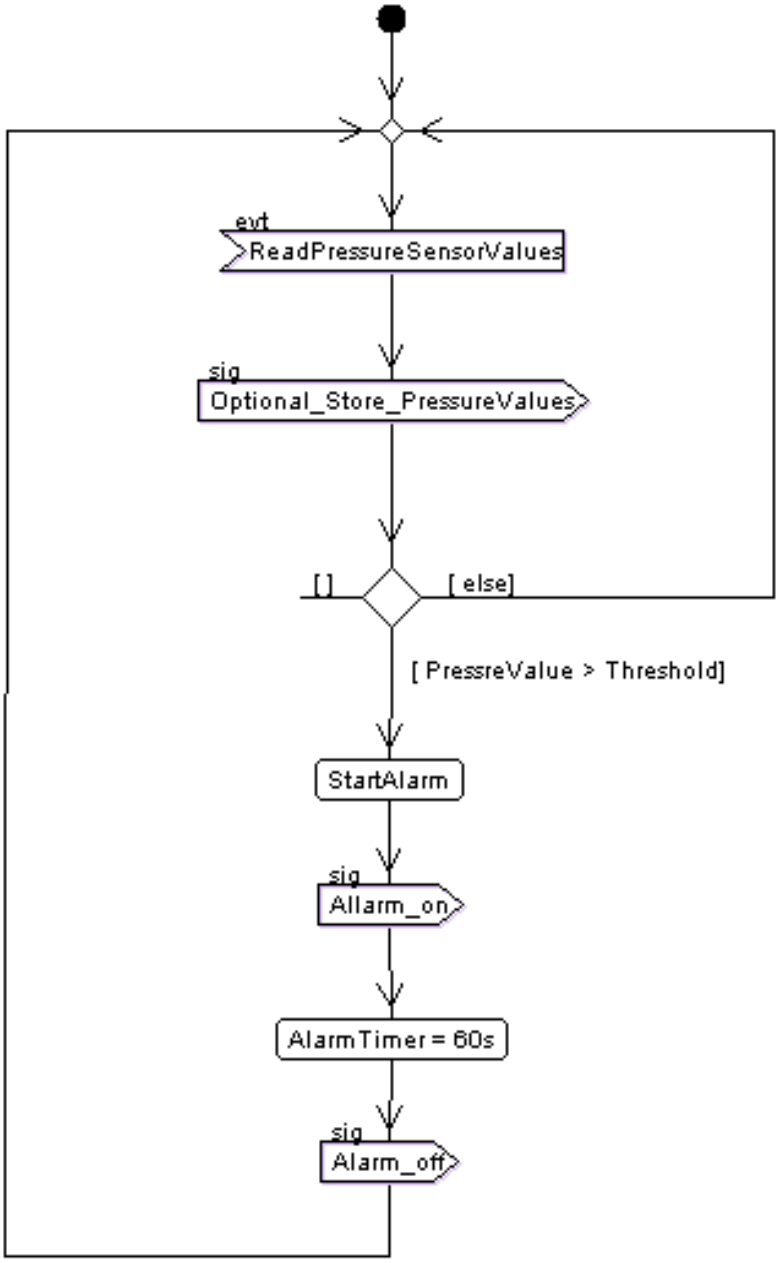


System Analysis

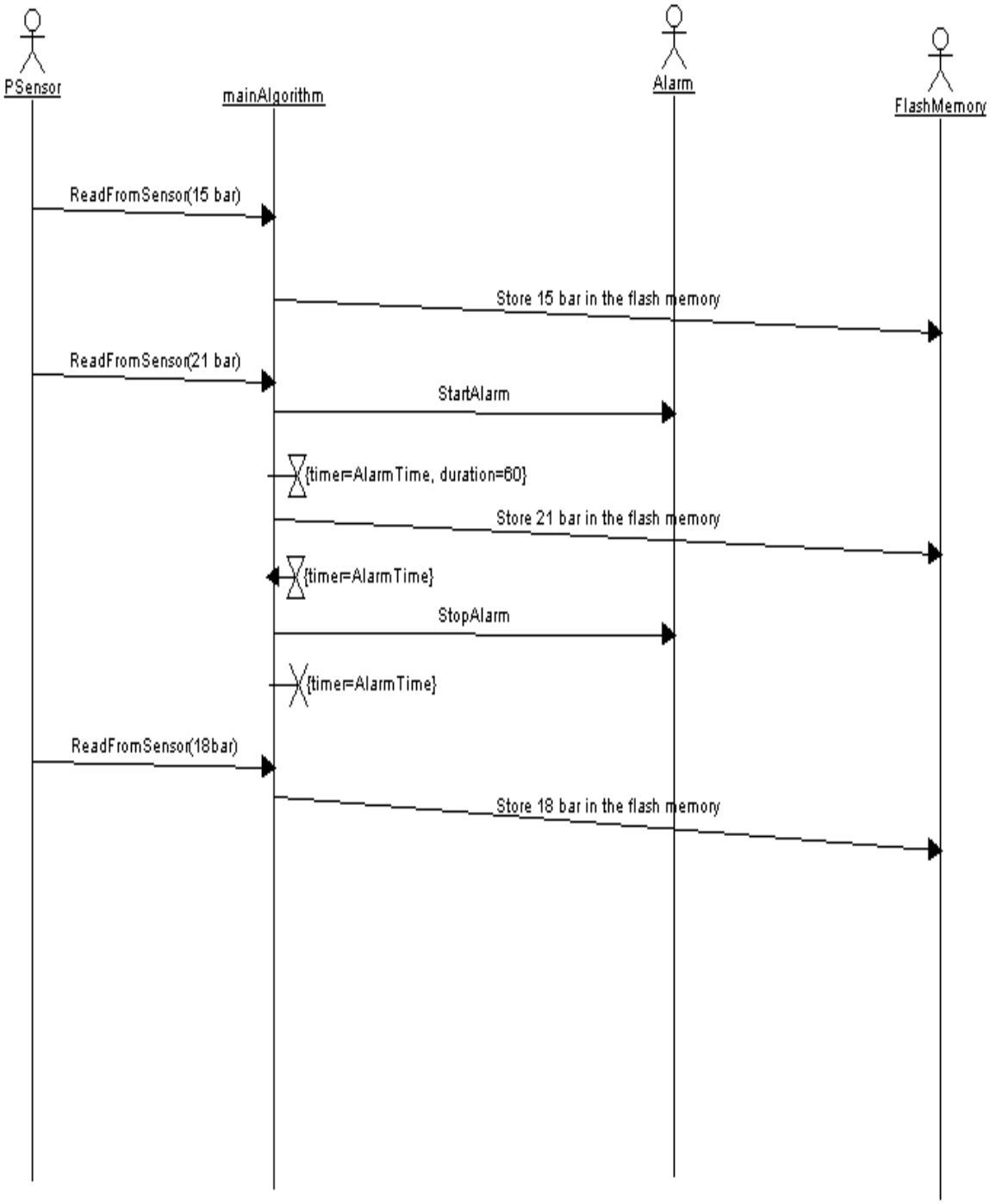
1. Usecase Diagram



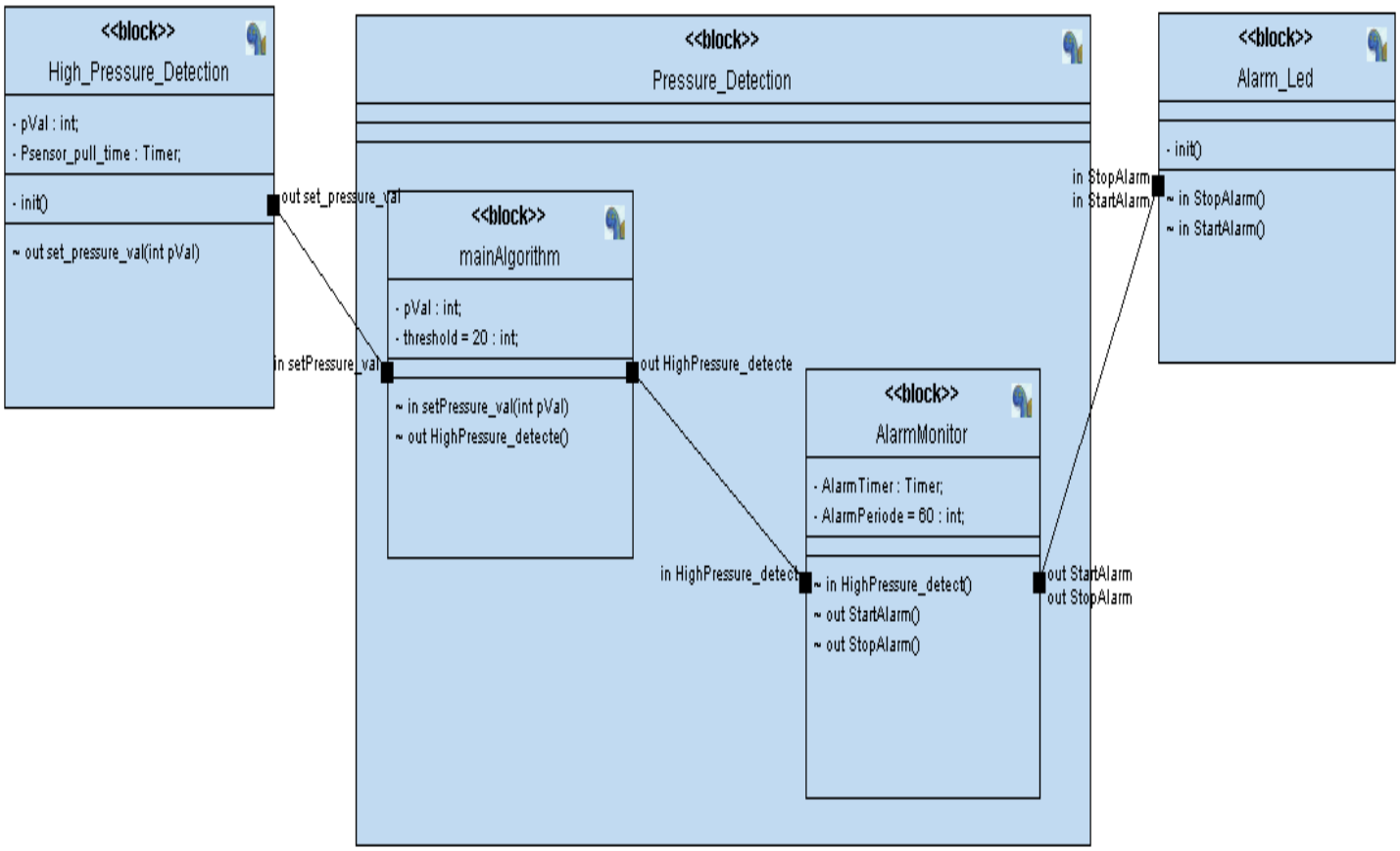
2. Activity Diagram



3. Sequence Diagram

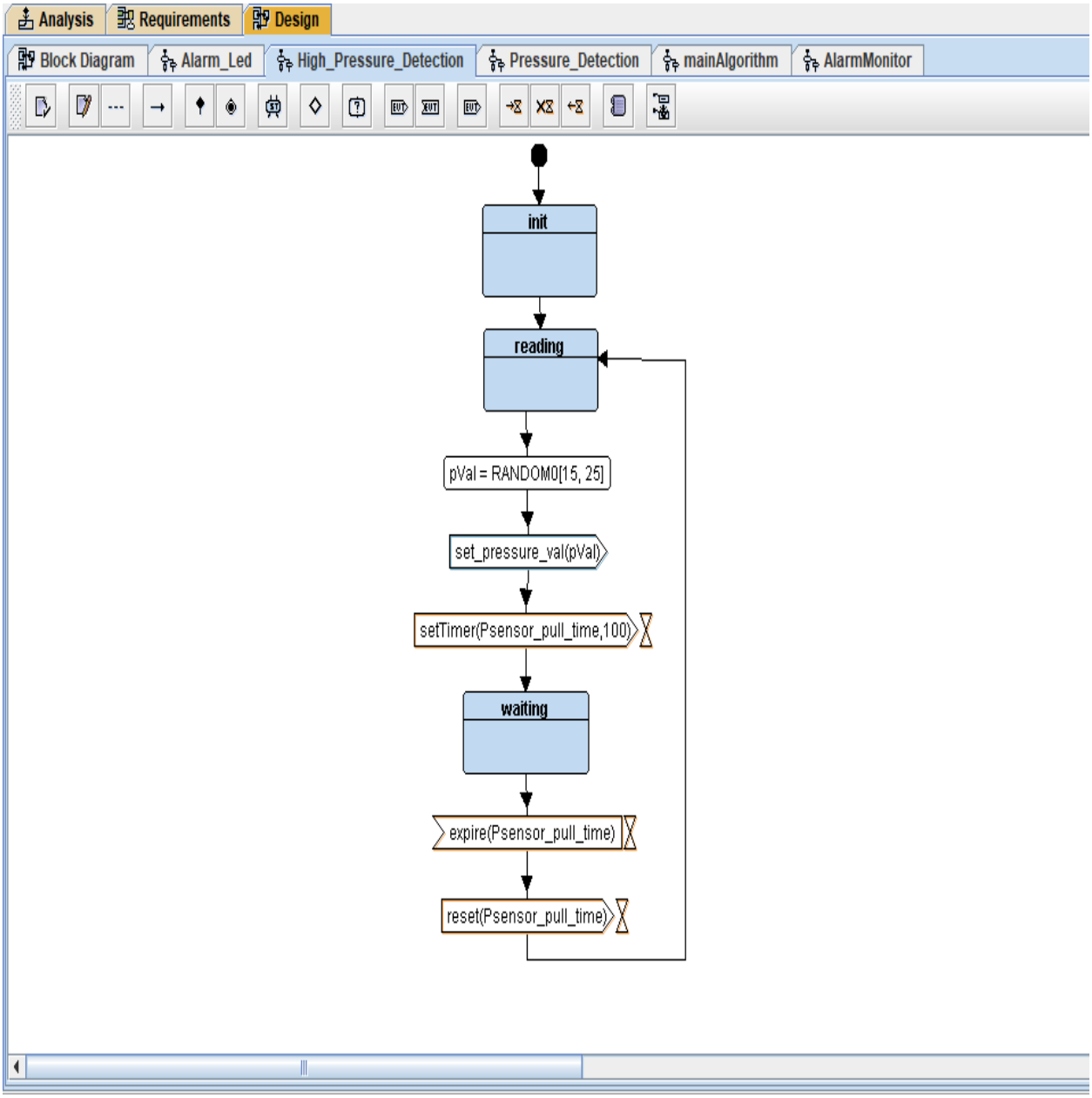


System Design Diagram

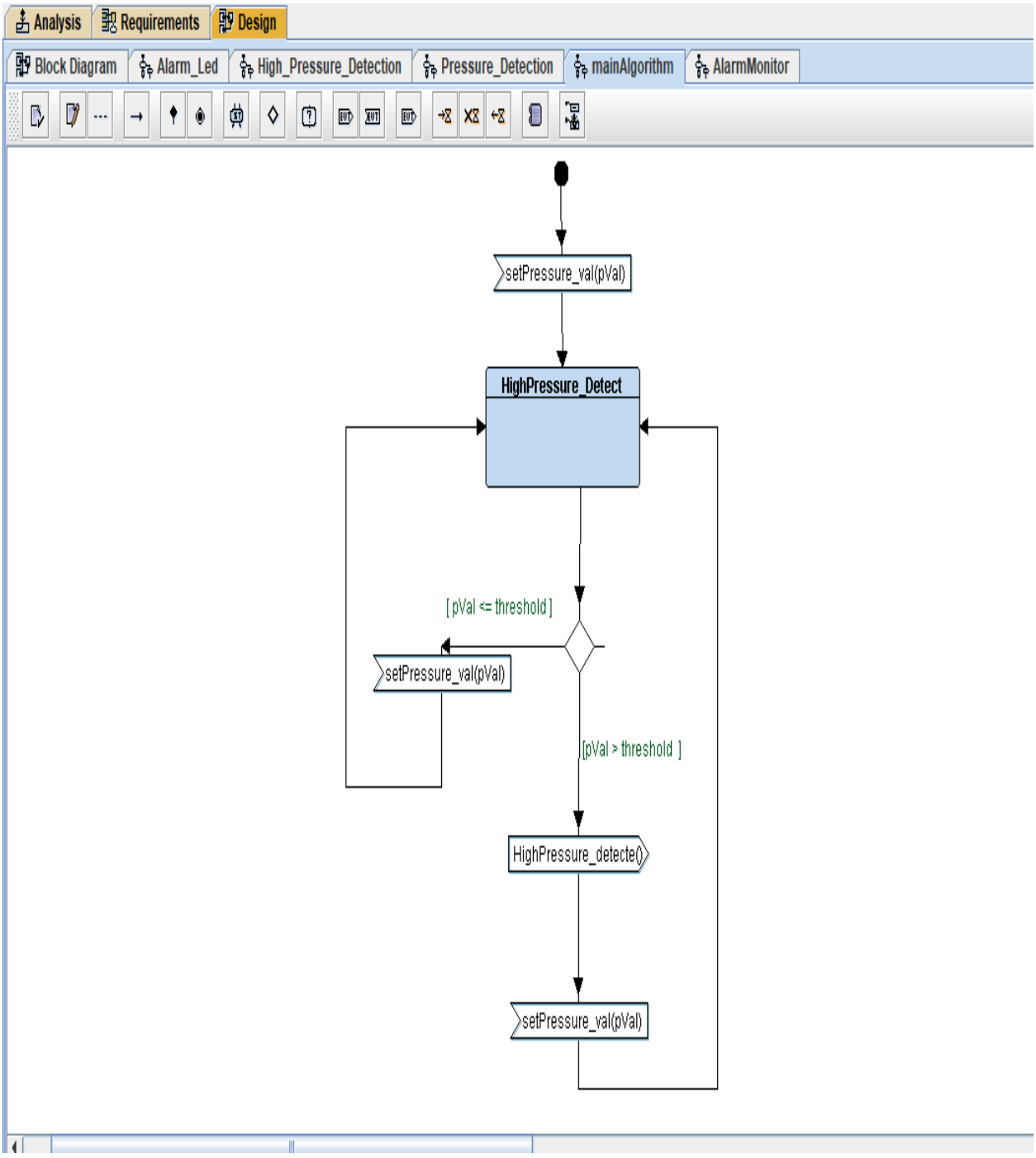


State Machine of Design Diagram

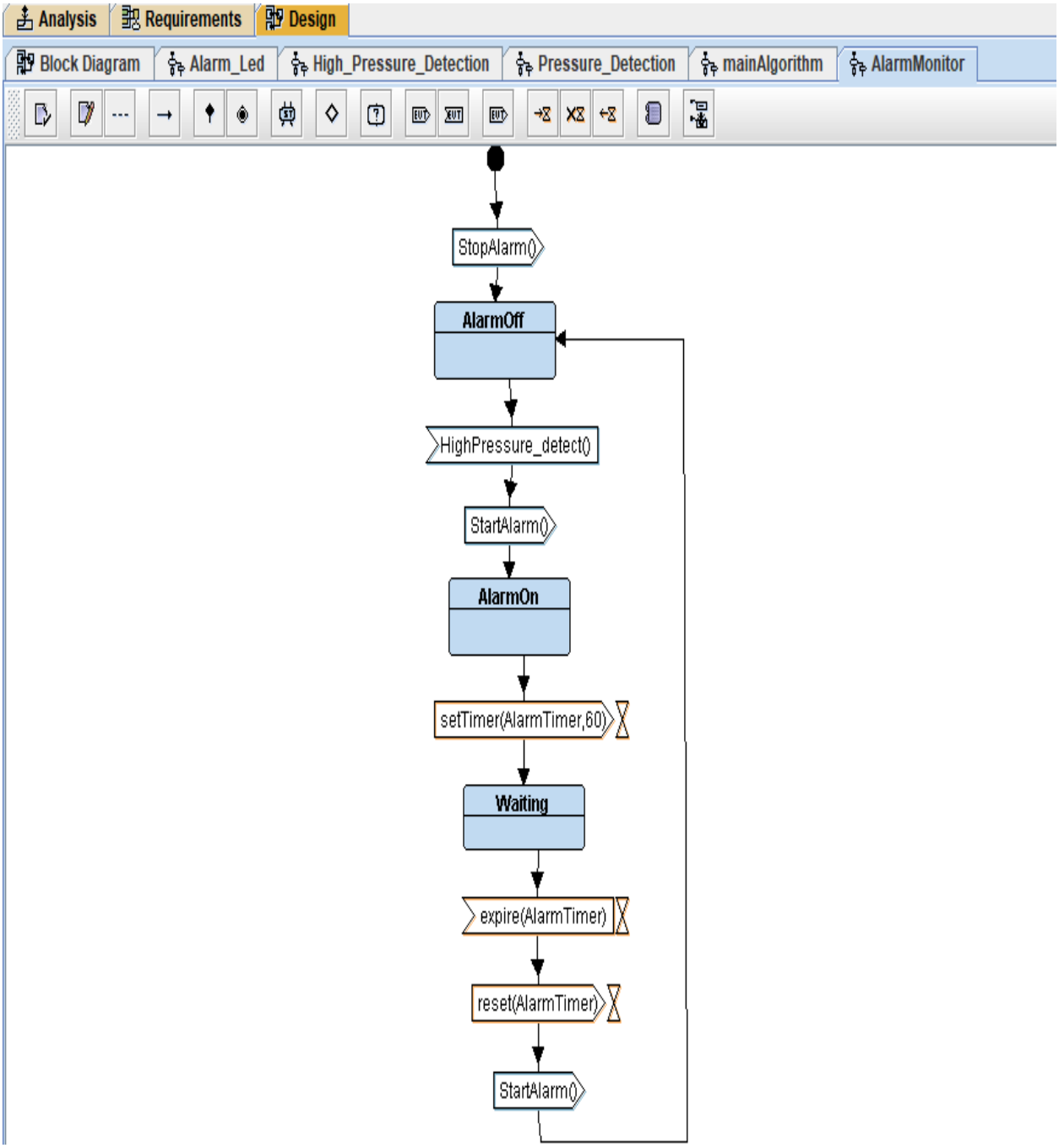
1. High_Pressure_Detection



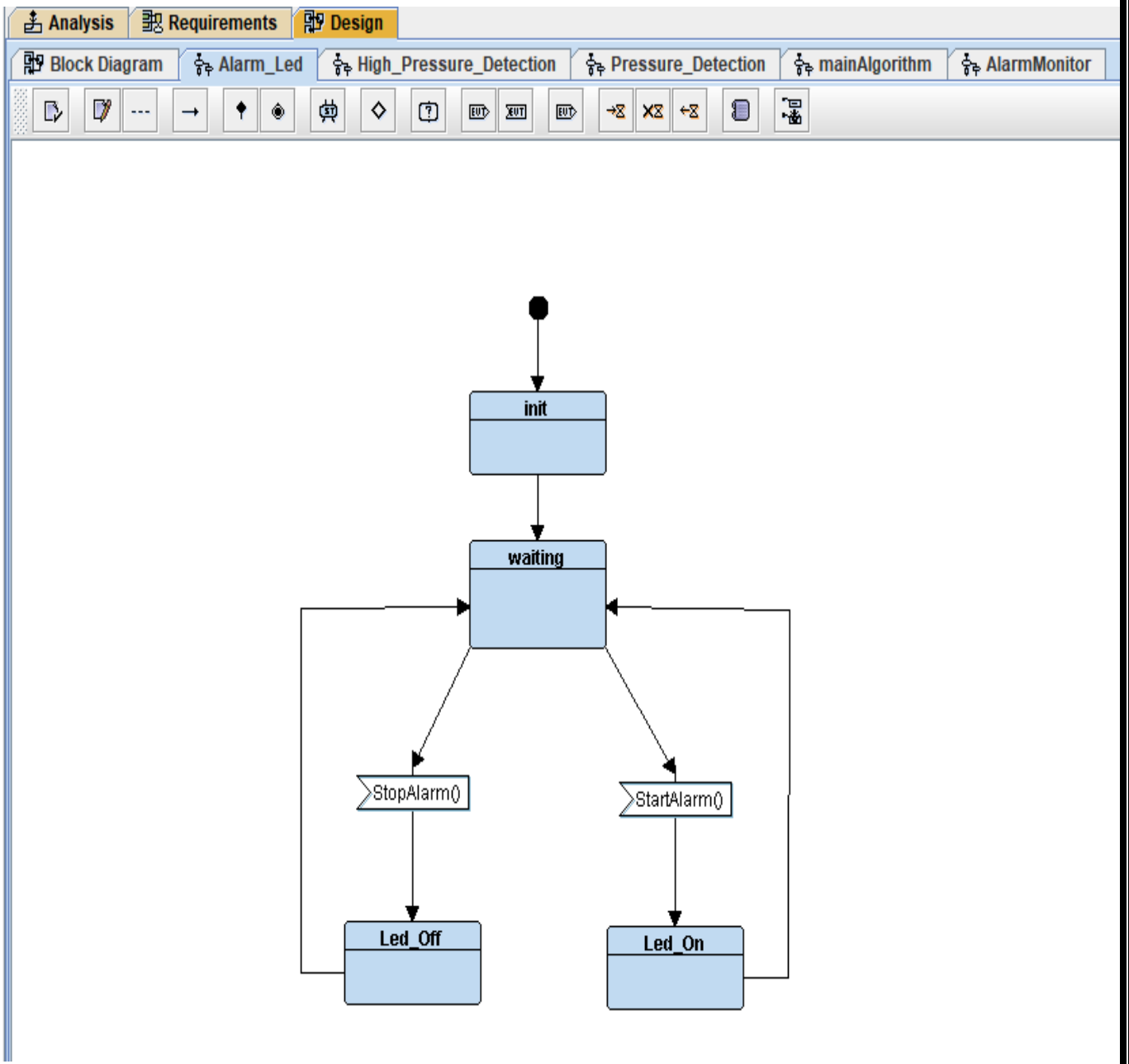
2. Main_Algorithm



3. Alarm_Monitor



4. Alarm_Acuator(led)



Files

1. State.h

```c
/*
 * state.h
 *
 *  Created on: 19 Jul 2024
 *      Author: Ahmed Basem
 */

#ifndef STATE_H_
#define STATE_H_

#include <stdio.h>
#include <stdlib.h>

//Automatic STATE Function generated
#define STATE_define(_statFUN_) void ST_##_statFUN_()
#define STATE(_statFUN_)  ST_##_statFUN_

// States Connection (interfaces between moduels)
/* implementation of these functions be in the destination */
void set_pressure_val (int pValue);
void high_pressure_detected();
void StartAlarm();
void StopAlarm();


#endif /* STATE_H_ */
```

2. Main.c

```c
/*
 * MainAlg.c
 *
 *  Created on: 19 Jul 2024
 *      Author: Ahmed Basem
 */

#include "MainAlg.h"


//variables
static int MainAlg_pValue = 0;
static int MainAlg_threshold = 20;

//STATE Pointer to function
void (*MainAlg_state)();


// interface between this moduel and PressureSensorDriver
void set_pressure_val (int pValue)
{
    MainAlg_pValue = pValue;
}
STATE_define(MainAlg_high_pressure_detection)
{
    //state_Name
    MainAlg_state_id = MainAlg_high_pressure_detection ;

    //state_Action
    if(MainAlg_pValue > MainAlg_threshold)
        high_pressure_detected();
}
```

3. Driver.h

```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)
#define GPIOA_BSRR *(volatile uint32_t *)(GPIO_PORTA + 0x10)
#define GPIOA_BRR *(volatile uint32_t *)(GPIO_PORTA + 0x14)


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

4. Driver.c

driver.c

```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;
}
```

5. mainAlgorith.h

```h
/*
 * MainAlg.h
 *
 *  Created on: 19 Jul 2024
 *      Author: Ahmed Basem
 */


#ifndef MAINALG_H_
#define MAINALG_H_

#include "state.h"

//Define States
enum {
    MainAlg_high_pressure_detection
}MainAlg_state_id;


//declare states functions MainAlg
STATE_define(MainAlg_high_pressure_detection);


//STATE Pointer to function
extern void (*MainAlg_state)();

#endif /* MAINALG_H_ */
```

6. mainAlgoritm.c

```c
/*
 * MainAlg.c
 *
 *  Created on: 19 Jul 2024
 *      Author: Ahmed Basem
 */

#include "MainAlg.h"


//variables
static int MainAlg_pValue = 0;
static int MainAlg_threshold = 20;

//STATE Pointer to function
void (*MainAlg_state)();


// interface between this moduel and PressureSensorDriver
void set_pressure_val (int pValue)
{
    MainAlg_pValue = pValue;
}
STATE_define(MainAlg_high_pressure_detection)
{
    //state_Name
    MainAlg_state_id = MainAlg_high_pressure_detection ;

    //state_Action
    if(MainAlg_pValue > MainAlg_threshold)
        high_pressure_detected();
}
```

7. highpressre_detction.h

```h
/*
 * PressureSensorDriver.
 *
 *  Created on: 19 Jul 2024
 *      Author: Ahmed Basem
 */

#ifndef PressureSensorDriver_H_
#define PressureSensorDriver_H_

#include "state.h"

//Define States
enum {
    PressureSensorDriver_reading
}PressureSensorDriver_state_id;


//declare states functions PressureSensorDriver
STATE_define(PressureSensorDriver_reading);

//methods
void PressureSensorDriver_init();


//STATE Pointer to function
extern void (*PressureSensorDriver_state)();

#endif /* PressureSensorDriver_H_ */
```

8. highpressre_detction.c

```c
/*
 * PressureSensorDriver.c
 *
 *  Created on: 19 Jul 2024
 *      Author: Ahmed Basem
 */

#include "PressureSensorDriver.h"
#include "driver.h"

//variables
static int PressureSensorDriver_pValue = 0;

//STATE Pointer to function
void (*PressureSensorDriver_state)();

void PressureSensorDriver_init()
{
    //init PressureSensorDriver
    GPIO_INITIALIZATION();
}
STATE_define(PressureSensorDriver_reading)
{
    //state_Name
    PressureSensorDriver_state_id = PressureSensorDriver_reading ;

    //state_Action
    PressureSensorDriver_pValue = getPressureVal();
    set_pressure_val(PressureSensorDriver_pValue); //send signal
    Delay(10000); //100 sec
}
```

9. MonitorAlarm.h

```c
/*
 * AlarmMonitor.h
 *
 *  Created on: 19 Jul 2024
 *      Author: Ahmed Basem
 */

#ifndef ALARMMONITOR_H_
#define ALARMMONITOR_H_

#include "state.h"

//Define States
enum {
    AlarmMonitor_alarmOff,
    AlarmMonitor_alarmOn
}AlarmMonitor_state_id;


//declare states functions AlarmMonitor
STATE_define(AlarmMonitor_alarmOff);
STATE_define(AlarmMonitor_alarmOn);

//STATE Pointer to function
extern void (*AlarmMonitor_state)();

#endif /* ALARMMONITOR_H_ */
```

10. MonitorAlarm.c

```c
#include "AlarmMonitor.h"
#include "driver.h"

//variables
static int AlarmMonitor_period = 60000;


//STATE Pointer to function
void (*AlarmMonitor_state)();


// interface between this moduel and MainAlg
void high_pressure_detected()
{
    AlarmMonitor_state = STATE(AlarmMonitor_alarmOn);
}

STATE_define(AlarmMonitor_alarmOff)
{
    //state_Name
    AlarmMonitor_state_id = AlarmMonitor_alarmOff ;

    //state_Action
    StopAlarm();
}
STATE_define(AlarmMonitor_alarmOn)
{
    //state_Name
    AlarmMonitor_state_id = AlarmMonitor_alarmOn ;

    //state_Action
    StartAlarm();
    Delay(AlarmMonitor_period); //60 sec
    StopAlarm();
    AlarmMonitor_state = STATE(AlarmMonitor_alarmOff);
}
```

11. AlarmLed.h

```c
/*
 * AlarmActuatorDriver.h
 *
 *  Created on: 19 Jul 2024
 *      Author: Ahmed Basem
 */

#ifndef AlarmActuatorDriver_H_
#define AlarmActuatorDriver_H_

#include "state.h"

//Define States
enum {
    AlarmActuatorDriver_waiting,
    AlarmActuatorDriver_AlarmOn,
    AlarmActuatorDriver_AlarmOff
}AlarmActuatorDriver_state_id;


//declare states functions AlarmActuatorDriver
STATE_define(AlarmActuatorDriver_waiting);
STATE_define(AlarmActuatorDriver_AlarmOn);
STATE_define(AlarmActuatorDriver_AlarmOff);

//methods
void AlarmActuatorDriver_init();

//STATE Pointer to function
extern void (*AlarmActuatorDriver_state)();
#endif /* AlarmActuatorDriver_H_ */
```

12. AlarmLed.c

```c
#define ALARM_OFF 1
#define ALARM_ON  0

//STATE Pointer to function
void (*AlarmActuatorDriver_state)();

void AlarmActuatorDriver_init()
{
    GPIO_INITIALIZATION();
}

// interface between this moduel and AlarmMonitor
void StartAlarm()
{
    AlarmActuatorDriver_state = STATE(AlarmActuatorDriver_AlarmOn);
    AlarmActuatorDriver_state();
}
void StopAlarm()
{
    AlarmActuatorDriver_state = STATE(AlarmActuatorDriver_AlarmOff);
    AlarmActuatorDriver_state();
}

STATE_define(AlarmActuatorDriver_waiting)
{
    //state_Name
    AlarmActuatorDriver_state_id = AlarmActuatorDriver_waiting ;
}

STATE_define(AlarmActuatorDriver_AlarmOn)
{
    //state_Name
    AlarmActuatorDriver_state_id = AlarmActuatorDriver_AlarmOn ;
    //state_Action
    Set_Alarm_actuator(ALARM_ON);
    AlarmActuatorDriver_state = STATE(AlarmActuatorDriver_waiting);
}

STATE_define(AlarmActuatorDriver_AlarmOff)
{
    //state_Name
    AlarmActuatorDriver_state_id = AlarmActuatorDriver_AlarmOff ;
    //state_Action
    Set_Alarm_actuator(ALARM_OFF);
    AlarmActuatorDriver_state = STATE(AlarmActuatorDriver_waiting);
}
```

13. Startup.c

```c
void Reset_Handler();
extern int main(void);

void Default_Handler()
{
    Reset_Handler();
}
void NMI_Handler() __attribute__((weak, alias ("Default_Handler")));;
void H_fault_Handler()__attribute__((weak, alias ("Default_Handler")));;

//reserve stack size
static unsigned long Stack_top[256]; //256*b = 1024B

void (*const g_p_func_Vectors[])() __attribute__((section(".vectors")))=
{
    (void (*)())((unsigned long)Stack_top + sizeof(Stack_top)),
    &Reset_Handler,
    &NMI_Handler,
    &H_fault_Handler
};

extern unsigned int _E_text;
extern unsigned int _S_DATA;
extern unsigned int _E_DATA;
extern unsigned int _S_bss;
extern unsigned int _E_bss;

void Reset_Handler()
{
    //copy data from ROM to Ram
    unsigned int DATA_size=(unsigned char*)&_E_DATA -(unsigned char*)&_S_DATA;
    unsigned char* P_src = (unsigned char*)&_E_text;
    unsigned char* P_dst = (unsigned char*)&_S_DATA;
    for(int i = 0; i < DATA_size; i++)
    {
        *((unsigned char*)P_dst++) = *((unsigned char*)P_src++);
    }
    //init the .bss with zero
    unsigned int bss_size=(unsigned char*)&_E_bss -(unsigned char*)&_S_bss;
    P_dst = (unsigned char*)&_S_bss;
    for(int i = 0; i < bss_size; i++)
    {
        *((unsigned char*)P_dst++) = (unsigned char)0;
    }
    //jumb to main();
```

14. Linker_script.ld

```
/*Linker_script Cortex_m4
Eng:Ahmed Basem
*/
MEMORY
{
    flash(RX) : ORIGIN = 0x08000000, LENGTH = 128k
    sram(RWX) : ORIGIN = 0X20000000, LENGTH = 30k
}
SECTIONS
{
    .text : {
            *(.vectors*)
            *(.text*)
            *(.rodata)
            _E_text = . ;
    }> flash

    .data : {
            _S_DATA = . ;
            *(.data)
            . = ALIGN(4);
            _E_DATA = . ;

    }> sram AT> flash

    .debug :{
            *(.debug*)
    }
    .bss : {
            _S_bss = . ;
            *(.bss*)
            _E_bss = . ;
    }> sram
}
```

15. Makefile

Makefile

```
#@copyright : Ahmed Basem

CC=arm-none-eabi-
CFLAGS= -mcpu=cortex-m3 -gdwarf-2
INCS=-I .
LIBS=
SRC= $(wildcard *.c)
OBJ= $(SRC:.c=.o)
As = $(wildcard *.s)
AsOBJ = $(As:.s=.o)
Project_name=PressureDetectionSystem


all: $(Project_name).bin
    @echo "======= Buid is Done ========"

%.o: %.c
    $(CC)gcc.exe -c $(CFLAGS) $(INCS) $< -o $@

%.s: %.c
    $(CC)gcc.exe -S $(CFLAGS) $(INCS) $< -o $@



$(Project_name).elf: $(AsOBJ) $(OBJ)
    $(CC)ld.exe -T linker_script.ld $(LIBS) $(AsOBJ) $(OBJ) -o $@ -Map=Map_file.map


$(Project_name).bin: $(Project_name).elf
    $(CC)objcopy.exe -O binary $< $@


clean-all:
    rm $(OBJ) $(Project_name).elf $(Project_name).bin

clean:
    rm $(Project_name).elf $(Project_name).bin
```

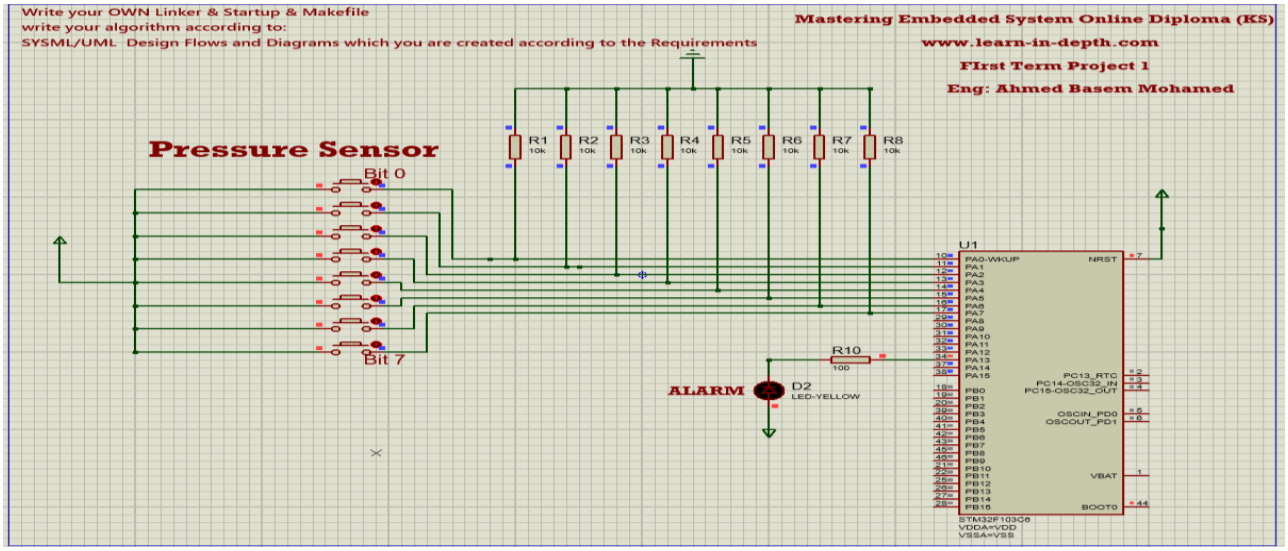
16. mapFile

```
                0x4             PressureSensorDriver.o
AlarmMonitor_state  0x4             AlarmMonitor.o
PressureSensorDriver_state_id
                0x1             PressureSensorDriver.o
MainAlg_state_id    0x1             MainAlg.o
AlarmActuatorDriver_state
                0x4             AlarmActuatorDriver.o
AlarmMonitor_state_id
                0x1             AlarmMonitor.o
AlarmActuatorDriver_state_id
                0x1             AlarmActuatorDriver.o

Memory Configuration

Name             Origin             Length             Attributes
flash            0x08000000         0x00020000         xr
sram             0x20000000         0x00007800         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map


.text           0x08000000      0x368
 *(.vectors*)
 .vectors       0x08000000       0x10 startup.o
                0x08000000                g_p_func_Vectors
 *(.text*)
 .text          0x08000010       0xac AlarmActuatorDriver.o
                0x08000010                AlarmActuatorDriver_init
                0x0800001c                StartAlarm
                0x08000038                StopAlarm
                0x08000054                ST_AlarmActuatorDriver_waiting
                0x0800006c                ST_AlarmActuatorDriver_AlarmOn
                0x08000094                ST_AlarmActuatorDriver_AlarmOff
 .text          0x080000bc       0x6c AlarmMonitor.o
                0x080000bc                high_pressure_detected
                0x080000d8                ST_AlarmMonitor_alarmOff
                0x080000f0                ST_AlarmMonitor_alarmOn
 .text          0x08000128       0x48 MainAlg.o
                0x08000128                set_pressure_val
                0x08000144                ST_MainAlg_high_pressure_detection
 .text          0x08000170       0x40 PressureSensorDriver.o
                0x08000170                PressureSensorDriver_init
                0x0800017c                ST_PressureSensorDriver_reading
 .text          0x080001b0       0xc4 driver.o
```

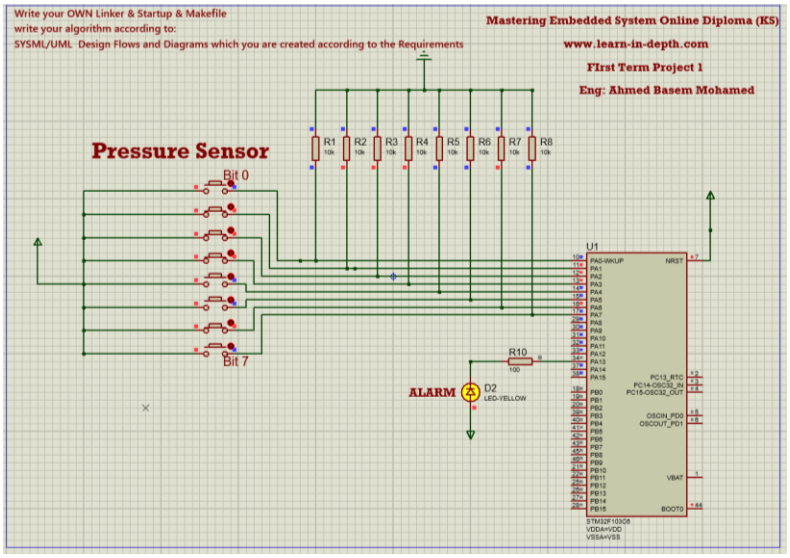
17. symbolTable

```
Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Unit_5/First Project/Code
$ arm-none-eabi-nm.exe FinalProject.elf
20000420 B _E_bss
20000018 D _E_DATA
08000400 T _E_text
20000018 B _S_bss
20000000 D _S_DATA
20000000 D alarmPeriod
2000000c D AMontor_state
08000370 T Default_Handler
08000188 T Delay
08000000 T g_p_func_Vectors
080001a8 T getPressureVal
080001fc T GPIO_INITIALIZATION
08000370 W H_fault_Handler
080000fc T highPressureDetected
20000008 D led_state
20000420 B led_State_iD
080002d4 T main
20000423 B main_State_id
20000010 D mAlgo_state
20000421 B monitorAlarm_id
08000370 W NMI_Handler
20000004 D p_state
2000001c B pressureVal
20000018 B pressureValue
20000422 B ps_state_id
0800037c T Reset_Handler
080001c0 T Set_Alarm_actuator
08000304 T setPressure
08000118 T ST_Alarm_off
08000130 T ST_Alarm_on
08000330 T ST_highPressure_state
08000048 T ST_led_init
080000c8 T ST_Led_off
08000094 T ST_Led_on
0800007c T ST_led_waiting
0800024c T ST_ps_init
08000270 T ST_reading
08000164 T ST_Waiting
080002ac T ST_WAiting
20000020 b Stack_top
08000010 T StartAlarm
0800002c T StopAlarm
20000014 D threshold

Osama@DESKTOP-BGUJ1JP MINGW64 /d/Embedded Diploma/Units/Unit_5/First Project/Code
$
```

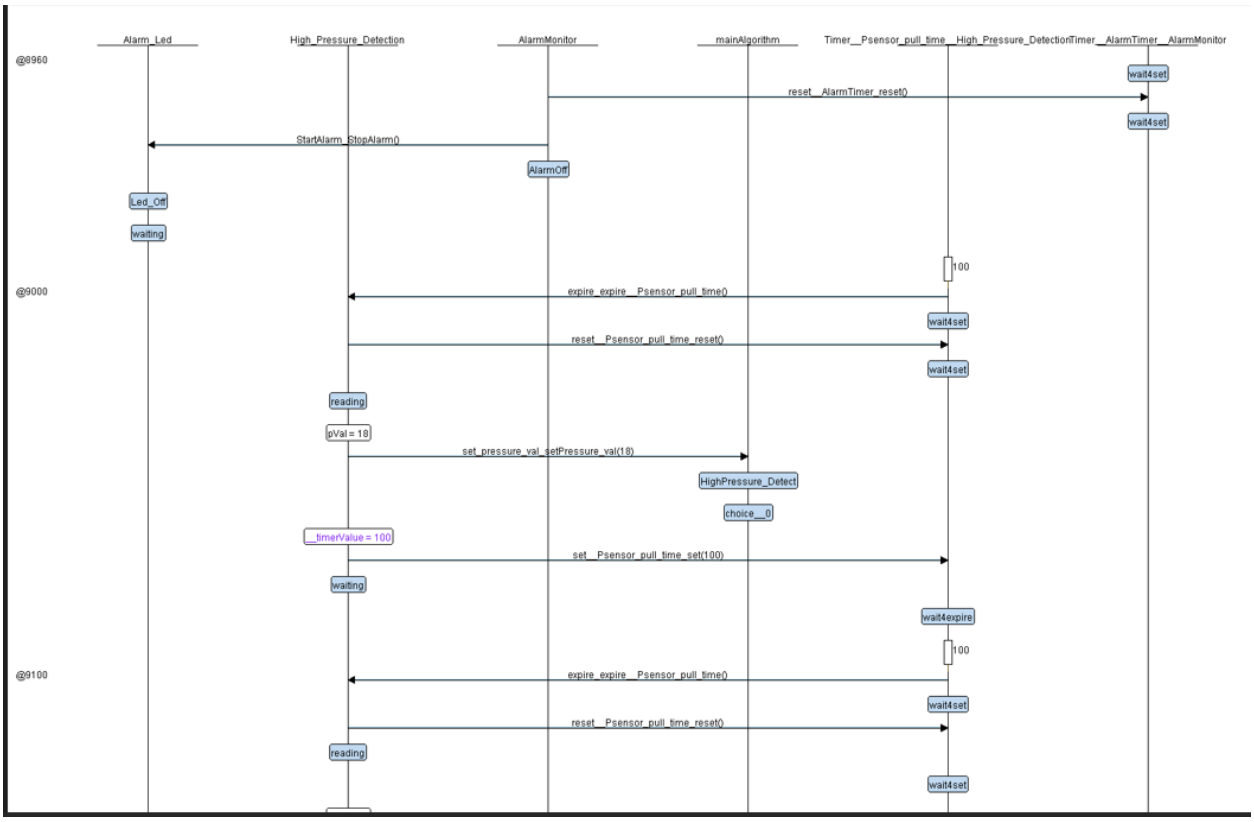
18. simulation before burn the bin file



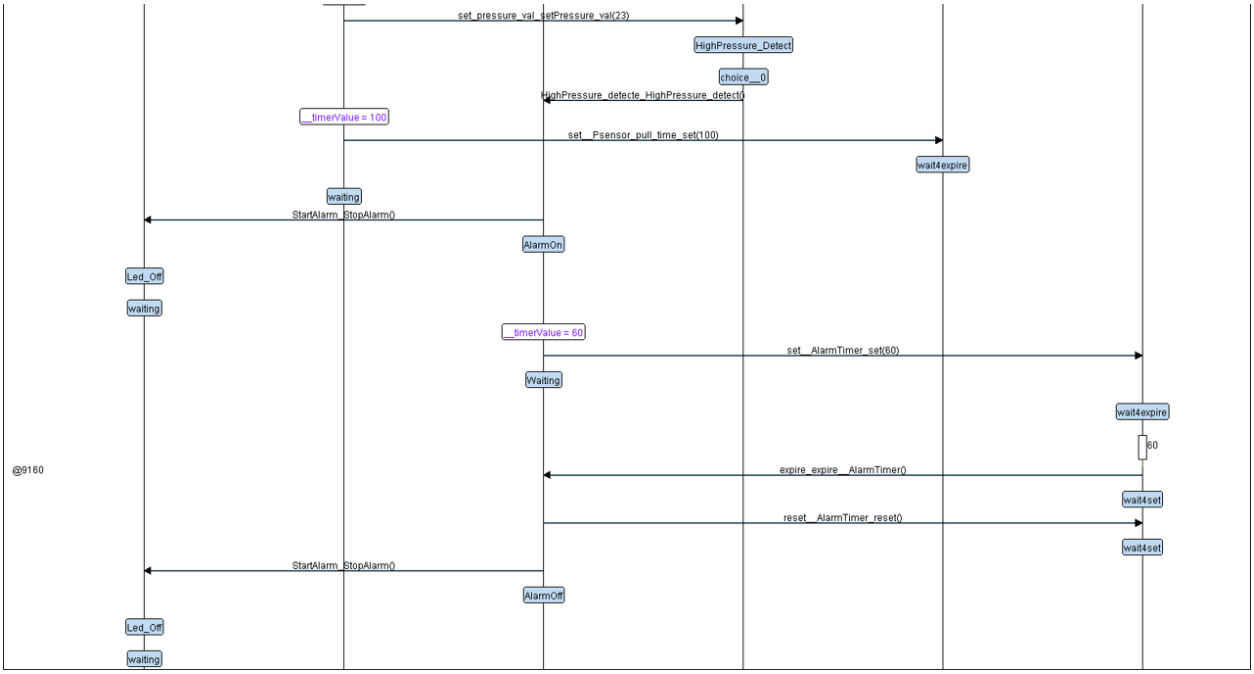
19. simulation after burn the bin file



20. interactive simulation1



21. interactive simulation2



22. interactive simulation3