Mastering Embedded Systems Diploma

https://www.learn-in-depth.com/

Submitted To : Eng/ Keroles Shenoda

S2S Automotive Solutions Technical Lead at Mentor Graphics

First Term Project 1

Case Study : Pressure Monitoring System Inside Airplane Cabin

Submitted By : Eng/ Peter Moner Goda

My Diploma Profile

GitHub Project

LinkedIn Profile

Case Study : Pressure Monitoring System Inside an Airplane Cabin

- Specifications :

  - If pressure value inside Airplane cabin is more than 20 bar >>> An Alarm System is going to turn on.
  - Alarm system stills ON for a 60 Second duration time .

- Assumption :

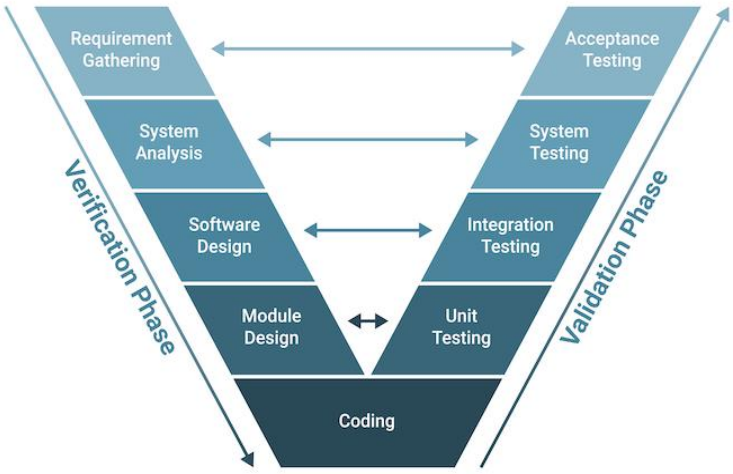
  - Pressure sensor never fails
  - Alarm never fails
  - The controller never faces power cutting
  - The controller setting up and shutting down are not modeled
  - The controller maintenance is not modeled

- Versioning :

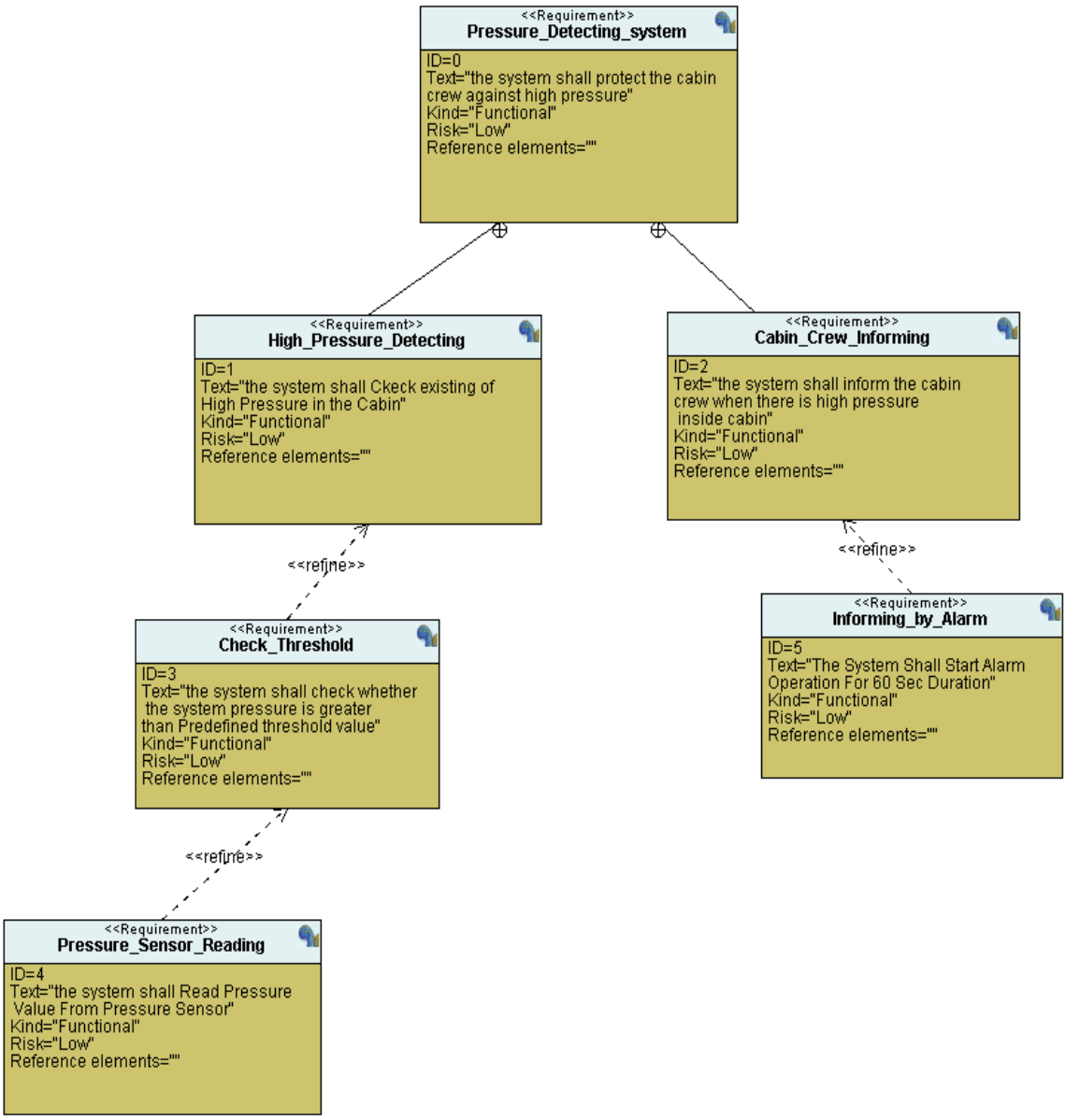
  - V1.0 ===> Output Product is a pressure monitoring system can informing cabin crew if there is a high pressure inside the cabin by turning Alarm system ON for 60 second Only.

- Method:

Using a V-model as the requirements and understanding of the software's functionality are well-defined from the beginning. The project scope is clear, and the development team has a solid understanding of the requirements.



- System Requirement:



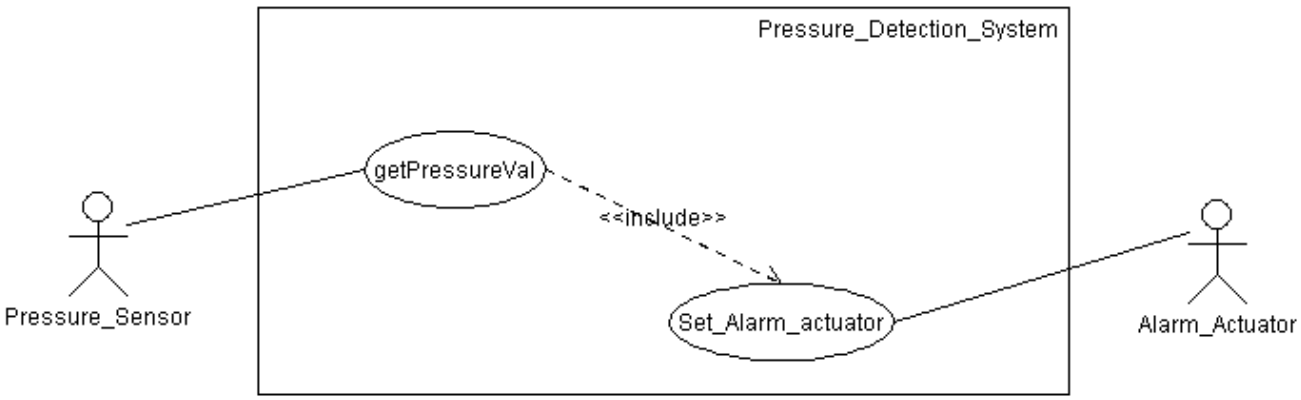
- Space Exploration and Partitioning

  - Selected Microcontrollers is **STM32F103C6**

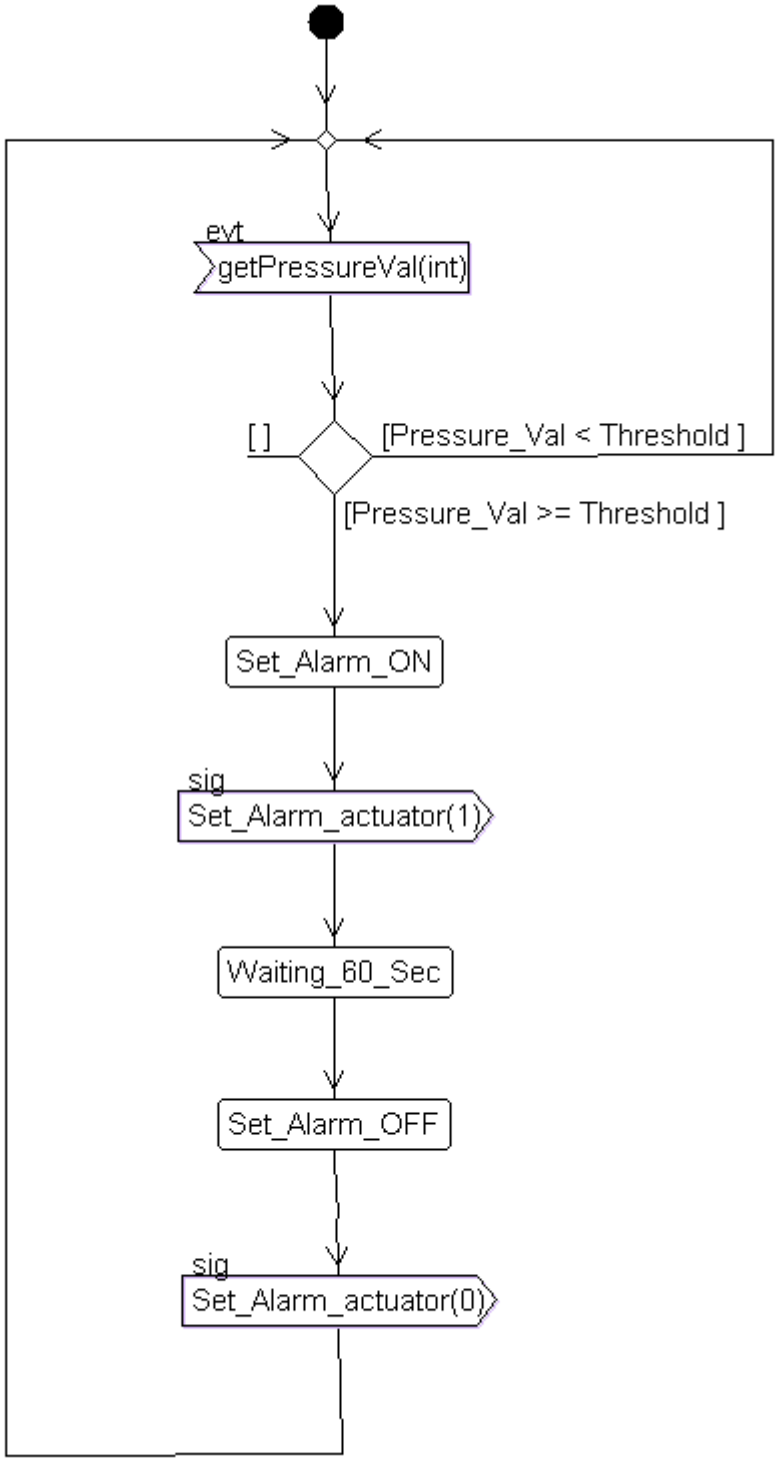
    - ARM 32-bit Cortex-M3 Microcontroller
    - Up to 72MHz Frequency
    - 32kB Flash
    - 10kB SRAM, PLL
    - Embedded Internal RC 8MHz and 32kHz
    - Real-Time Clock
    - Nested Interrupt Controller, Power Saving Modes, JTAG and SWD, 2 Synch.
    - 16-bit Timers with Input Capture, Output Compare and PWM, 16-bit 6-ch Advanced Timer, 16-bit Watchdog Timers, SysTick Timer
    - SPI, I2C, 2 USART, USB 2.0 Full Speed Interface, CAN 2.0B Active
    - 2 12-bit 10-ch A/D Converter
    - Fast I/O Ports.

- System Analysis:

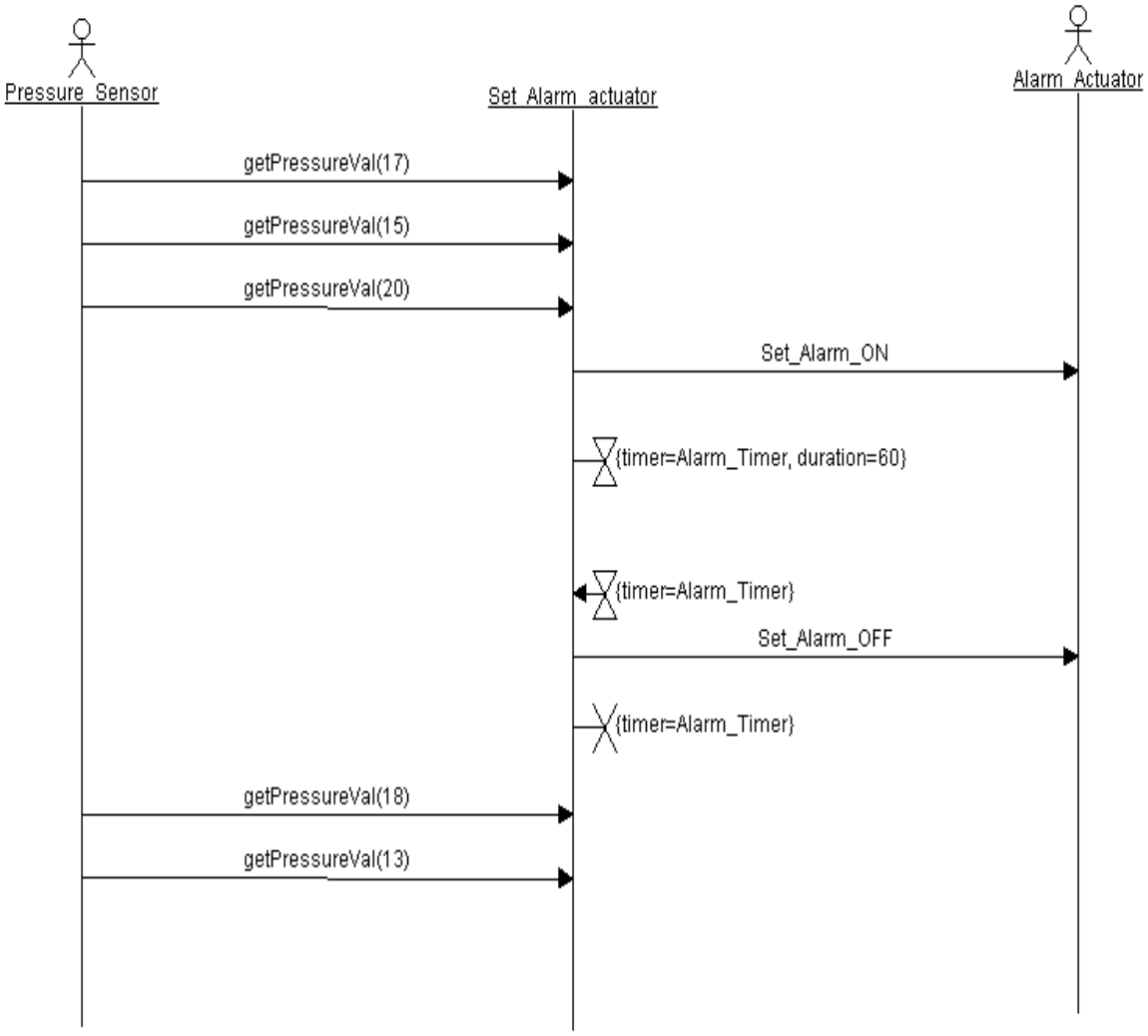
  - Case Diagram



- Activity Diagram

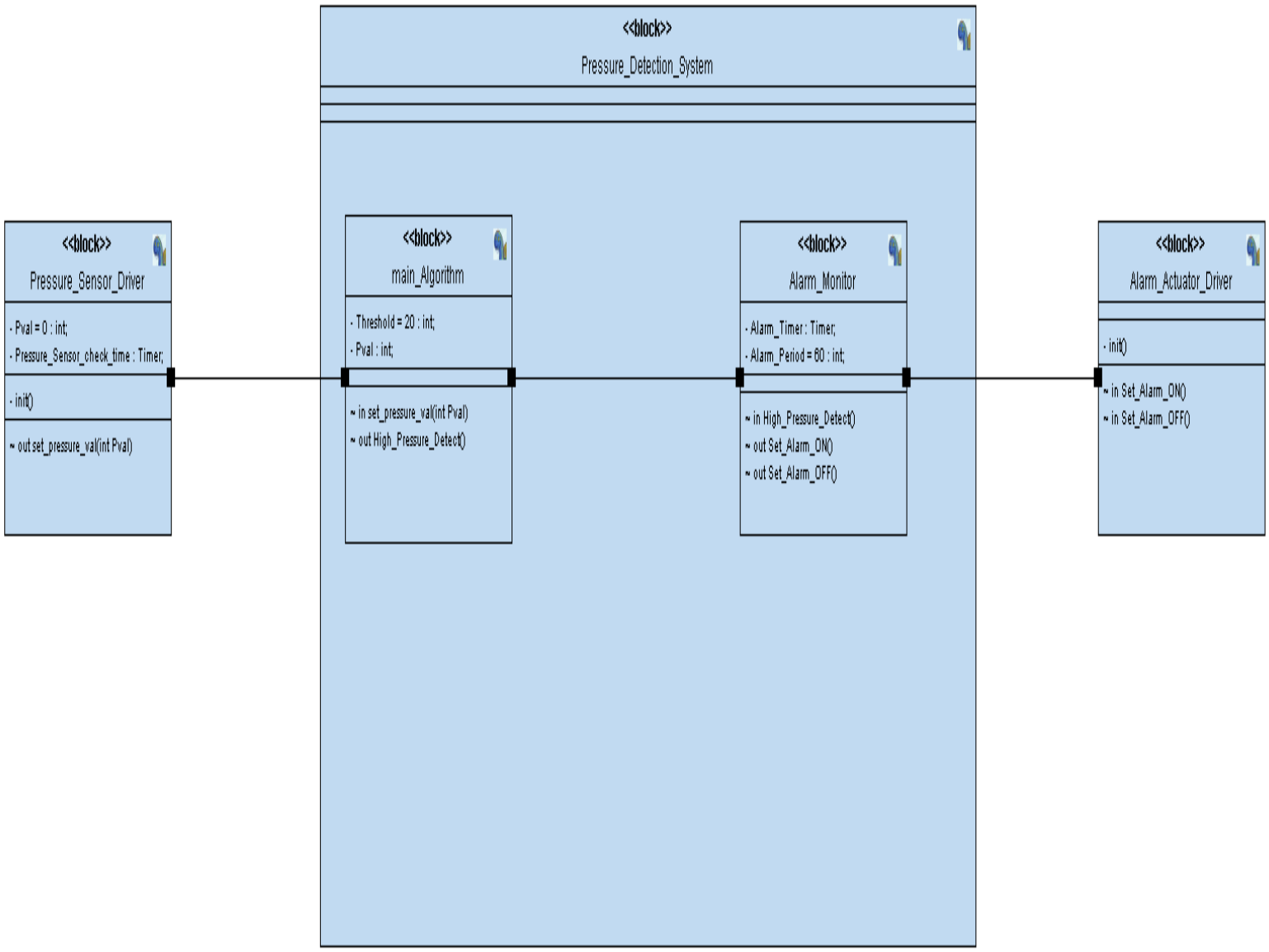


- Sequence Diagram

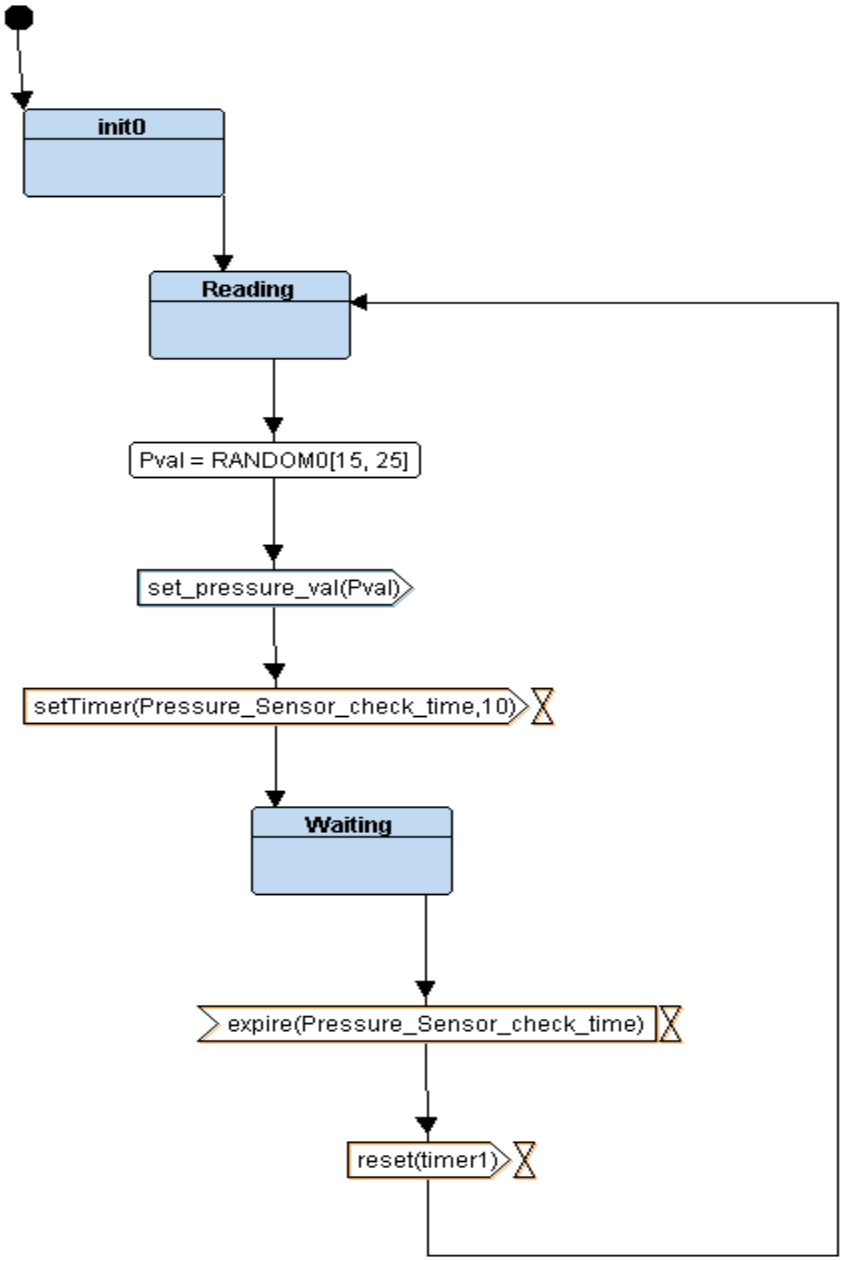


- System Design:

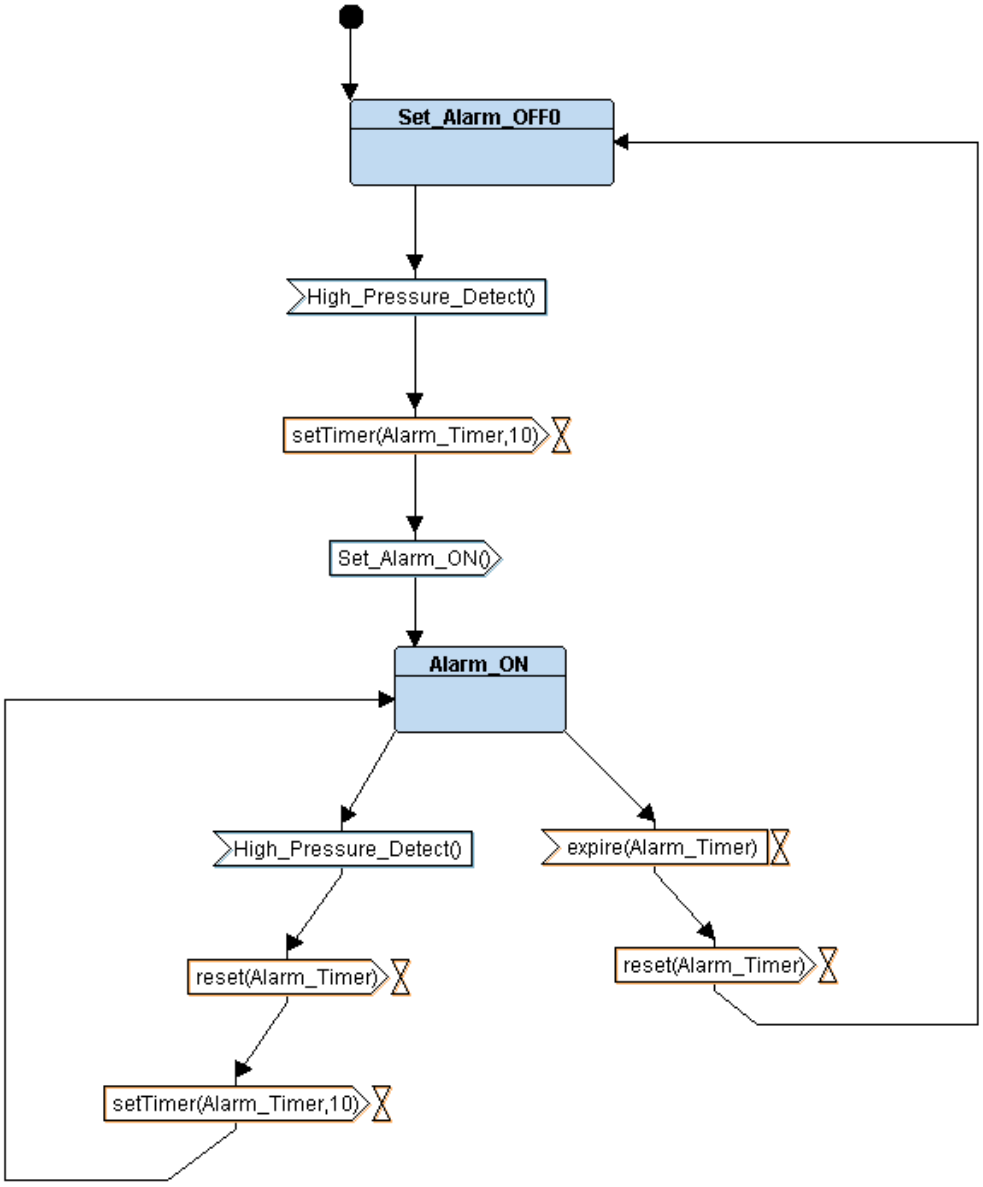
  - Full System Block Diagram



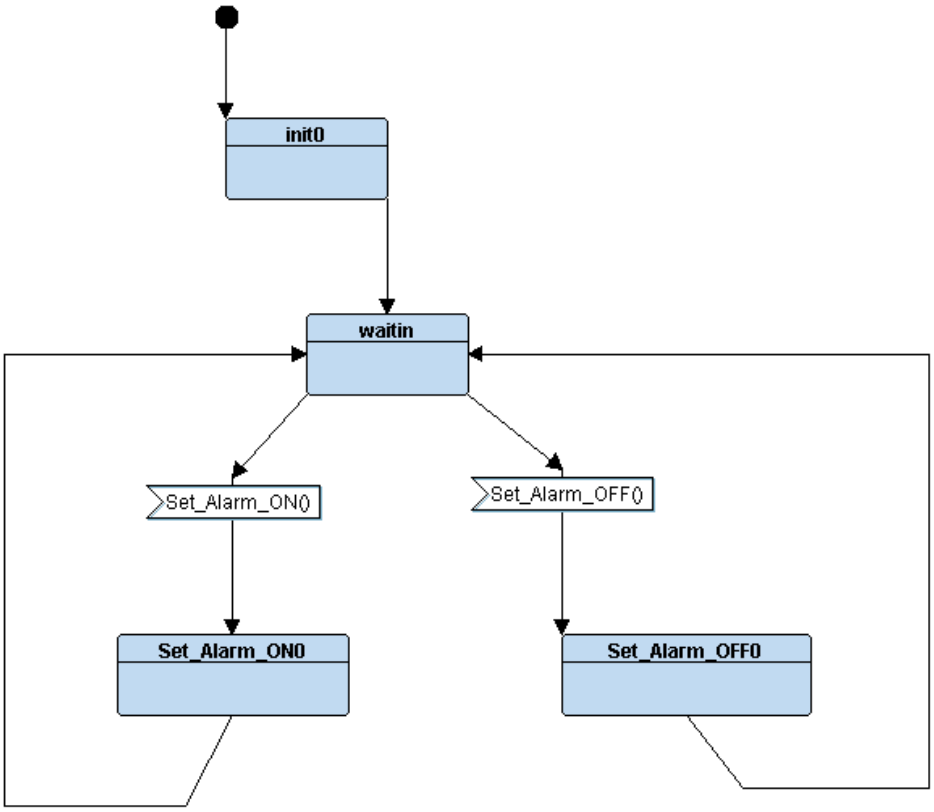
- State machine of Pressure Sensor Driver



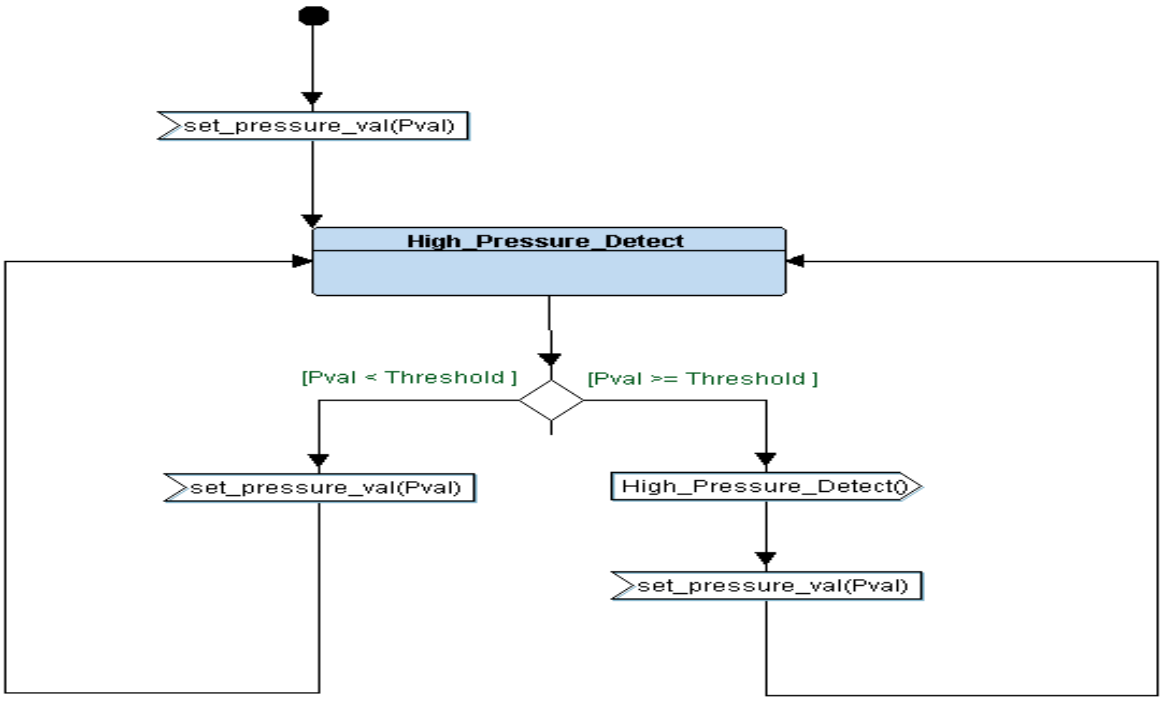
- State machine of Alarm Monitor



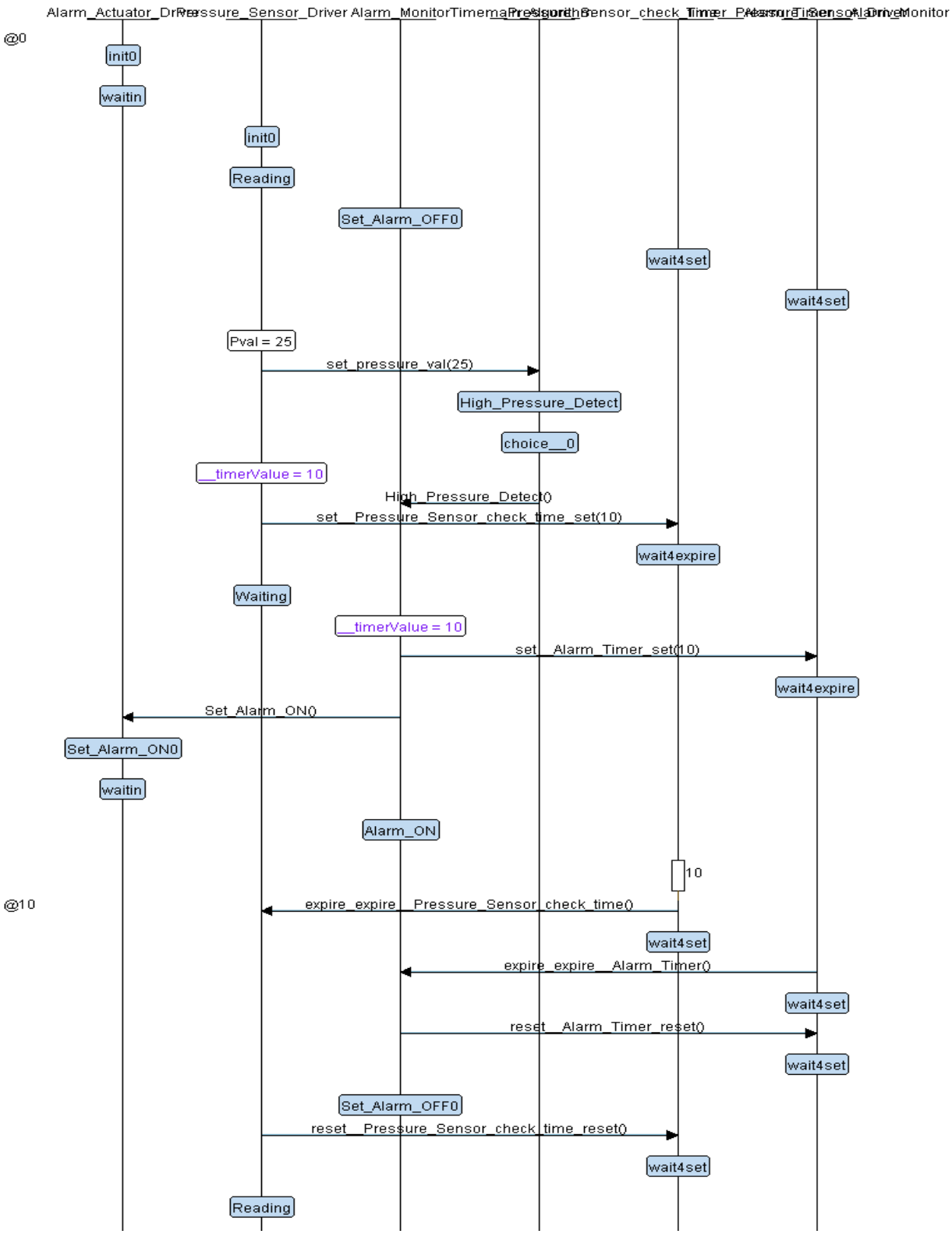
- State machine of Alarm System Driver

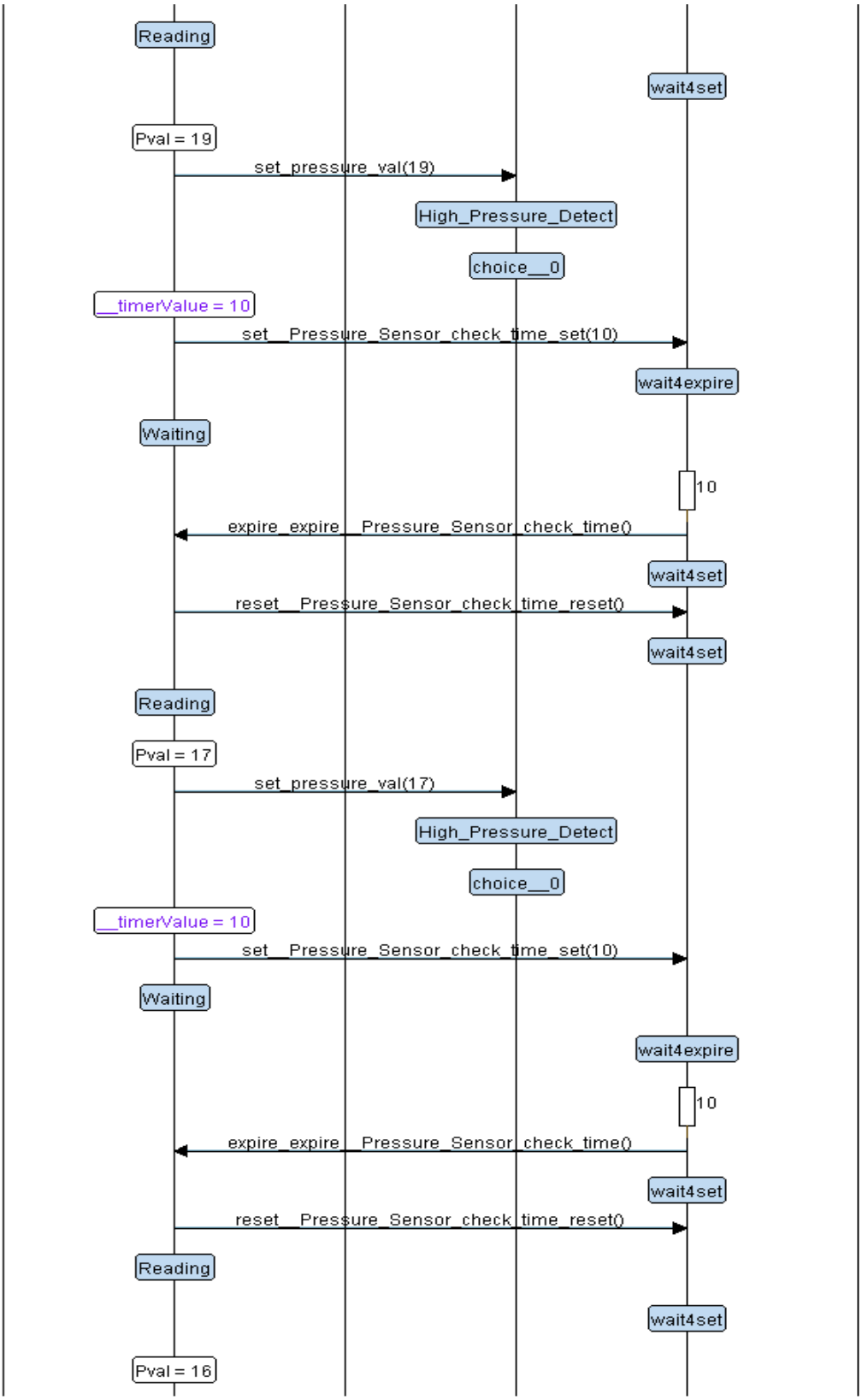


- State machine of Main Algorithm



- System Design Logic Simulation




Reading
wait4set
Pval = 19
set_pressure_val(19)
High_Pressure_Detect
choice__0
__timerValue = 10
set__Pressure_Sensor_check_time_set(10)
wait4expire
Waiting
10
@30
expire_expire__Pressure_Sensor_check_time()
wait4set
reset__Pressure_Sensor_check_time_reset()
wait4set
Reading
Pval = 17
set_pressure_val(17)
High_Pressure_Detect
choice__0
__timerValue = 10
set__Pressure_Sensor_check_time_set(10)
Waiting
wait4expire
10
@40
expire_expire__Pressure_Sensor_check_time()
wait4set
reset__Pressure_Sensor_check_time_reset()
Reading
wait4set
Pval = 16

- C Files

  - 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 == HIGH){
        RESET_BIT(GPIOA_ODR,13);
    }
    else if (i == LOW){
        SET_BIT(GPIOA_ODR,13);
    }
}

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

  - Main.c

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

#include "driver.h"
#include "Platform_Types.h"

#define Threshold_val 20

int main (){

    uint32 Pressure_value=0;

    GPIO_INITIALIZATION();
    Set_Alarm_actuator(LOW);

    while (1)
    {
        Pressure_value = getPressureVal();
        if(Pressure_value >= Threshold_val)
        {
            Set_Alarm_actuator(HIGH);
            Delay(1000000);

        }

        else{
            Set_Alarm_actuator(LOW);
        }
    }

}
```

- 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 HIGH 1
#define LOW  0

#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 ();
```

- Startup.c
- LinkerScript.ld
- Makefile

- Section Table

  - Driver.o

```
DELL@Peter-Moner MINGW32 /d/Embedded Systems K.S/UNIT 5/First_Ter
$ arm-none-eabi-objdump.exe -h driver.o

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   00000099  00000000  00000000  000003c8  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000012b  00000000  00000000  00000461  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  0000058c  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  0000059e  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000078  00000000  00000000  000005d4  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

  - Main.o

```
DELL@Peter-Moner MINGW32 /d/Embedded Systems K.S/UNIT 5/First_Term
$ arm-none-eabi-objdump.exe -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000048  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000007c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000007c  2**0
                  ALLOC
  3 .debug_info   000000ad  00000000  00000000  0000007c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 0000005e  00000000  00000000  00000129  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000038  00000000  00000000  00000187  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  000001bf  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000052  00000000  00000000  000001df  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000000f8  00000000  00000000  00000231  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  00000329  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  0000033b  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000030  00000000  00000000  00000370  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

- Startup.o

```
DELL@Peter-Moner MINGW32 /d/Embedded Systems K.S/UNIT 5/First_Term
$ arm-none-eabi-objdump.exe -h startup.o

startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000bc  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000f0  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000f0  2**0
                  ALLOC
  3 .vectors      0000001c  00000000  00000000  000000f0  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  4 .debug_info   00000182  00000000  00000000  0000010c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 000000cb  00000000  00000000  0000028e  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    00000064  00000000  00000000  00000359  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  000003bd  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_line   0000007b  00000000  00000000  000003dd  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  9 .debug_str    00000159  00000000  00000000  00000458  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .comment      00000012  00000000  00000000  000005b1  2**0
                  CONTENTS, READONLY
 11 .ARM.attributes 00000033  00000000  00000000  000005c3  2**0
                  CONTENTS, READONLY
 12 .debug_frame  0000004c  00000000  00000000  000005f8  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

- Pressure_Dection_System.elf

```
DELL@Peter-Moner MINGW32 /d/Embedded Systems K.S/UNIT 5/First_Term_
$ arm-none-eabi-objdump.exe -h Pressure_Detection_System.elf

Pressure_Detection_System.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000022c  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .debug_info   00000332  00000000  00000000  0000822c  2**0
                  CONTENTS, READONLY, DEBUGGING
  2 .debug_abbrev 000001c6  00000000  00000000  0000855e  2**0
                  CONTENTS, READONLY, DEBUGGING
  3 .debug_loc    00000164  00000000  00000000  00008724  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_aranges 00000060  00000000  00000000  00008888  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_line   00000166  00000000  00000000  000088e8  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_str    0000019b  00000000  00000000  00008a4e  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .comment      00000011  00000000  00000000  00008be9  2**0
                  CONTENTS, READONLY
  8 .ARM.attributes 00000033  00000000  00000000  00008bfa  2**0
                  CONTENTS, READONLY
  9 .debug_frame  000000f4  00000000  00000000  00008c30  2**2
                  CONTENTS, READONLY, DEBUGGING
```

- Symbol Table

  - Individual files symbol table

```
DELL@Peter-Moner MINGW32 /d/Embedded
$ arm-none-eabi-nm.exe  driver.o
00000000 T Delay
00000024 T getPressureVal
0000008c T GPIO_INITIALIZATION
0000003c T Set_Alarm_actuator

DELL@Peter-Moner MINGW32 /d/Embedded
$ arm-none-eabi-nm.exe  main.o
         U Delay
         U getPressureVal
         U GPIO_INITIALIZATION
00000000 T main
         U Set_Alarm_actuator

DELL@Peter-Moner MINGW32 /d/Embedded
$ arm-none-eabi-nm.exe  startup.o
         U _E_bss
         U _E_data
         U _E_text
         U _S_bss
         U _S_data
         U _stack_top
000000b0 W Bus_Fault_Handler
000000b0 T Default_Handler
000000b0 W Hard_Fault_Handler
         U main
000000b0 W MM_Fault_Handler
000000b0 W NMI_Handler
00000000 T Reset_Handler
000000b0 W Usage_Fault_Handler
00000000 D vectors
```

  - Pressure_Dection_System.elf

```
DELL@Peter-Moner MINGW32 /d/Embedded Systems K.S/UNIT 5/
$ arm-none-eabi-nm.exe  Pressure_Detection_System.elf
20000000 T _E_bss
20000000 T _E_data
0800022c T _E_rodata
0800022c T _E_text
20000000 T _S_bss
20000000 T _S_data
0800022c T _S_rodata
20001000 T _stack_top
080000cc W Bus_Fault_Handler
080000cc T Default_Handler
08000120 T Delay
08000144 T getPressureVal
080001ac T GPIO_INITIALIZATION
080000cc W Hard_Fault_Handler
080000d8 T main
080000cc W MM_Fault_Handler
080000cc W NMI_Handler
0800001c T Reset_Handler
0800015c T Set_Alarm_actuator
080000cc W Usage_Fault_Handler
08000000 T vectors
```

- Map File

```
Memory Configuration

Name             Origin             Length             Attributes
FLASH            0x08000000         0x00020000         xr
SRAM             0x20000000         0x00005000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map


.text           0x08000000      0x22c
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text)
 .text          0x0800001c       0xbc startup.o
                0x0800001c                Reset_Handler
                0x080000cc                MM_Fault_Handler
                0x080000cc                Bus_Fault_Handler
                0x080000cc                Default_Handler
                0x080000cc                Usage_Fault_Handler
                0x080000cc                Hard_Fault_Handler
                0x080000cc                NMI_Handler
 .text          0x080000d8       0x48 main.o
                0x080000d8                main
 .text          0x08000120      0x10c driver.o
                0x08000120                Delay
                0x08000144                getPressureVal
                0x0800015c                Set_Alarm_actuator
                0x080001ac                GPIO_INITIALIZATION
                0x0800022c                . = ALIGN (0x4)
                0x0800022c                _E_text = .
```

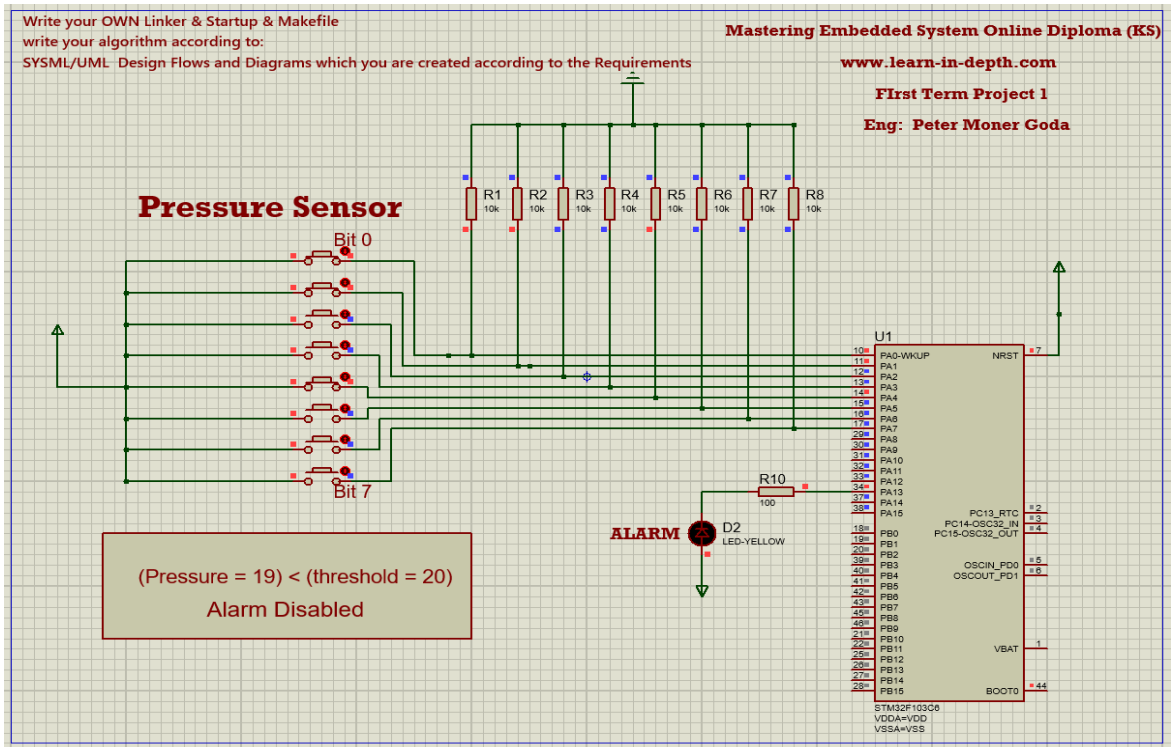
```
.data           0x20000000        0x0 load address 0x0800022c
                0x20000000                . = ALIGN (0x4)
                0x20000000                _S_data = .
 *(.data*)
 .data          0x20000000        0x0 startup.o
 .data          0x20000000        0x0 main.o
 .data          0x20000000        0x0 driver.o
                0x20000000                . = ALIGN (0x4)
                0x20000000                _E_data = .

.igot.plt       0x20000000        0x0 load address 0x0800022c
 .igot.plt      0x00000000        0x0 startup.o

.rodata         0x0800022c        0x0
                0x0800022c                . = ALIGN (0x4)
                0x0800022c                _S_rodata = .
 *(.rodata*)
                0x0800022c                . = ALIGN (0x4)
                0x0800022c                _E_rodata = .

.bss            0x20000000        0x0
                0x20000000                . = ALIGN (0x4)
                0x20000000                _S_bss = .
 *(.bss*)
 .bss           0x20000000        0x0 startup.o
 .bss           0x20000000        0x0 main.o
 .bss           0x20000000        0x0 driver.o
                0x20000000                . = ALIGN (0x4)
 *(COMMON*)
                0x20000000                . = ALIGN (0x4)
                0x20000000                _E_bss = .
                0x20000000                . = ALIGN (0x4)
                0x20001000                . = (. + 0x1000)
                0x20001000                _stack_top = .
```

- Hardware Simulation
  - AT pressure value < Predefined Threshold value



  - AT pressure value >= Predefined Threshold value