High Pressure Detection Project(1)

Name : Thomas Ashraf Zaki

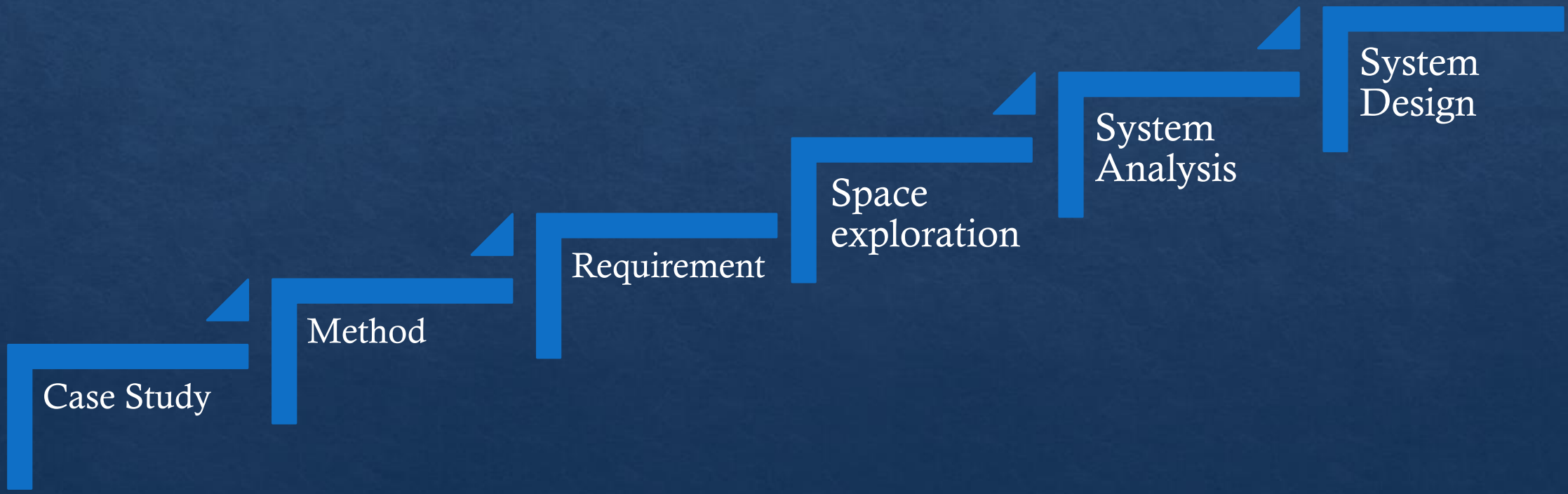
Profile : Click Here

GitHub : Click Here

Project Is

- Customer Want pressure controller That Turn On Alarm if The value of pressure Above 20 BAR
- By Make All this files using System architecting → (UML2)
- Without use any IDE to write the files
- Make the linker script , startup and makefile

Using Design Sequence



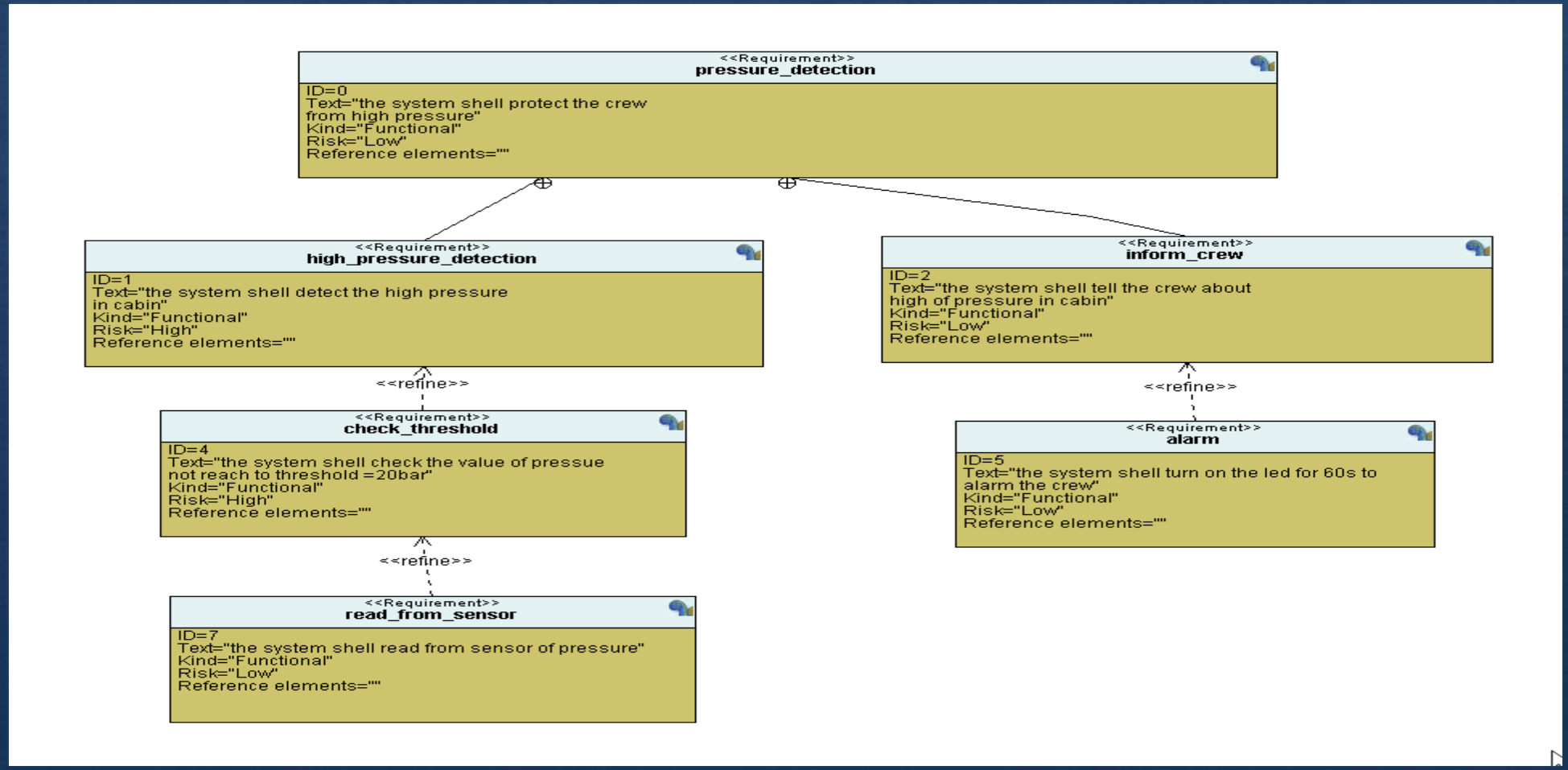
Case Study

- A "**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.

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

Requirement

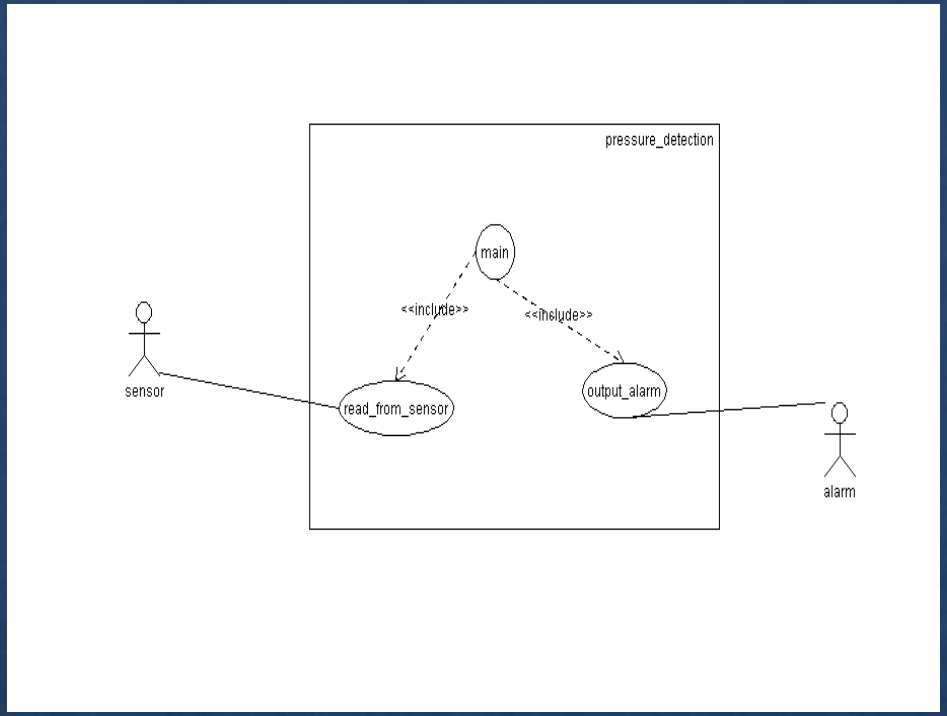


System Analysis

Analysis method

- System boundary and main functions →**Use Case Diagram**
- Relations between main functions →**Activity Diagram**
- Communications between main system entities and **actors** →**Sequence Diagram**

1-Use Case Diagram



Shows what the system does and who uses it

- 1.Define the boundary of the system

  Inside of the rectangle →**What you promise to design**

  Outside of the rectangle →**System environment (= Actors)**

This is not part of what you will have to design

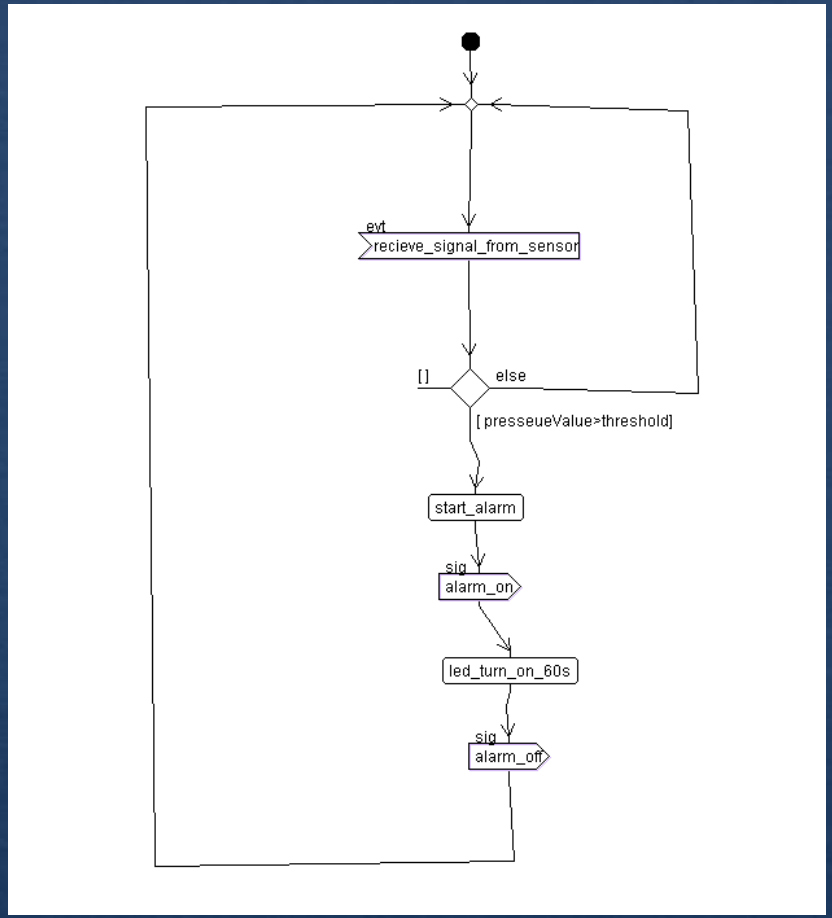
Name the system

- **1.Identify the services to be offered by the system**

Only services interacting with actors

- 2.Draw interactions between functions and actors

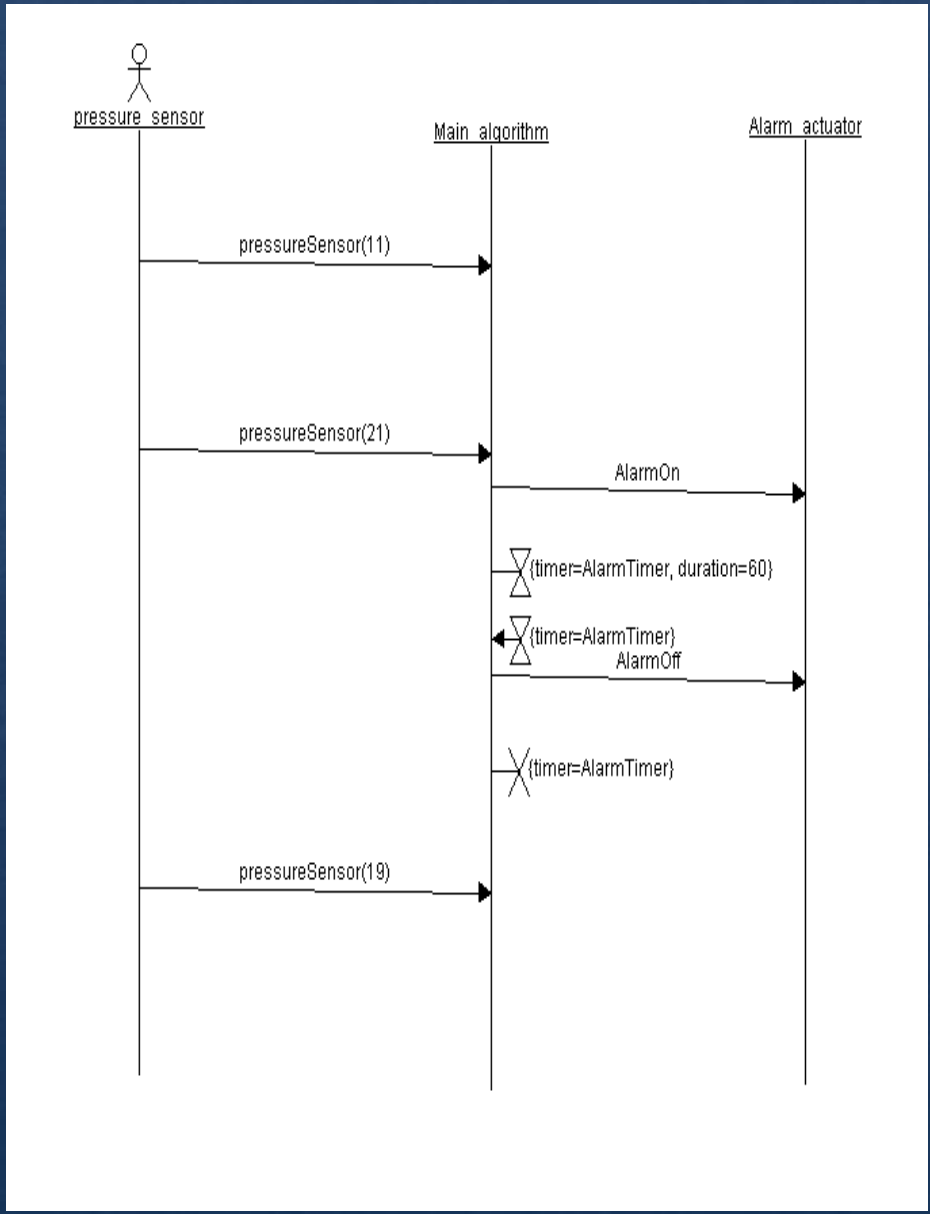
2-Activity Diagram



- Activity diagrams describe the workflow behaviour of a system
- An activity diagram is a special case of a state chart diagram in which states are activities ("functions")

3-Sequence Diagram

- An interaction diagram that details how operations are carried out.
- What messages are sent and when.
- Sequence diagrams are organized according to time
- NO message between actors
- One global clock (applies to the entire system)



System Design

- **Design = Making what a client wants**

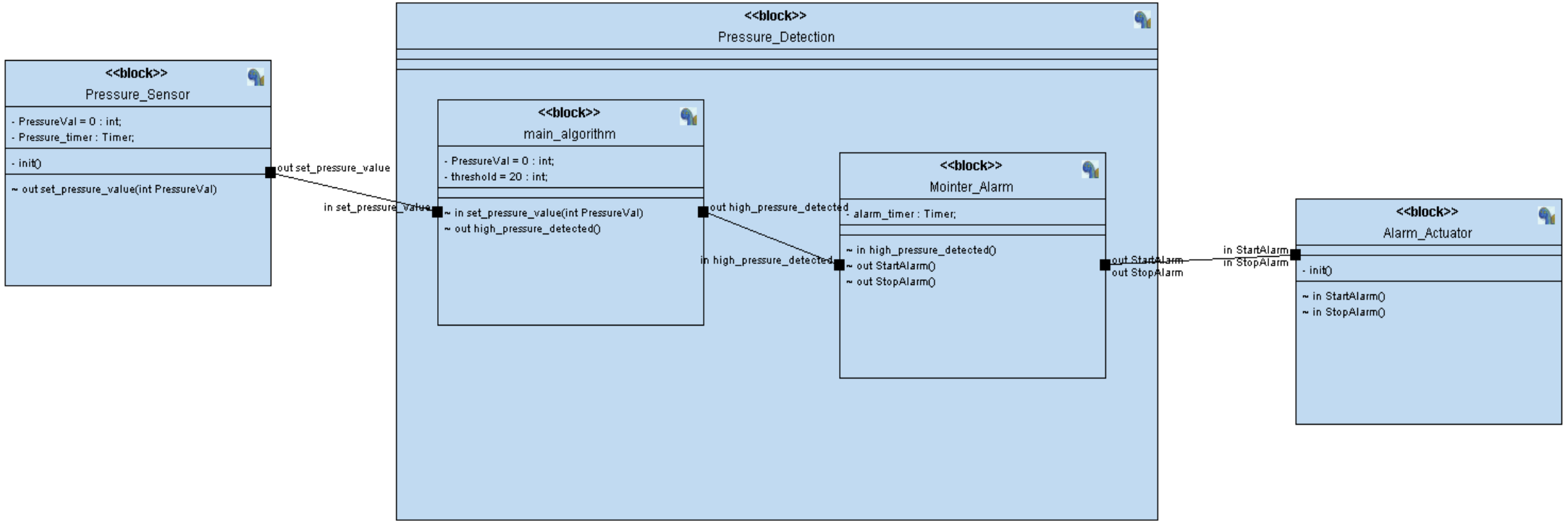
  ”creating a system” that complies with the client requirements.

System architecture →Block Definition Diagram and Internal Block Diagram

Behaviour of the system →State Machine Diagram

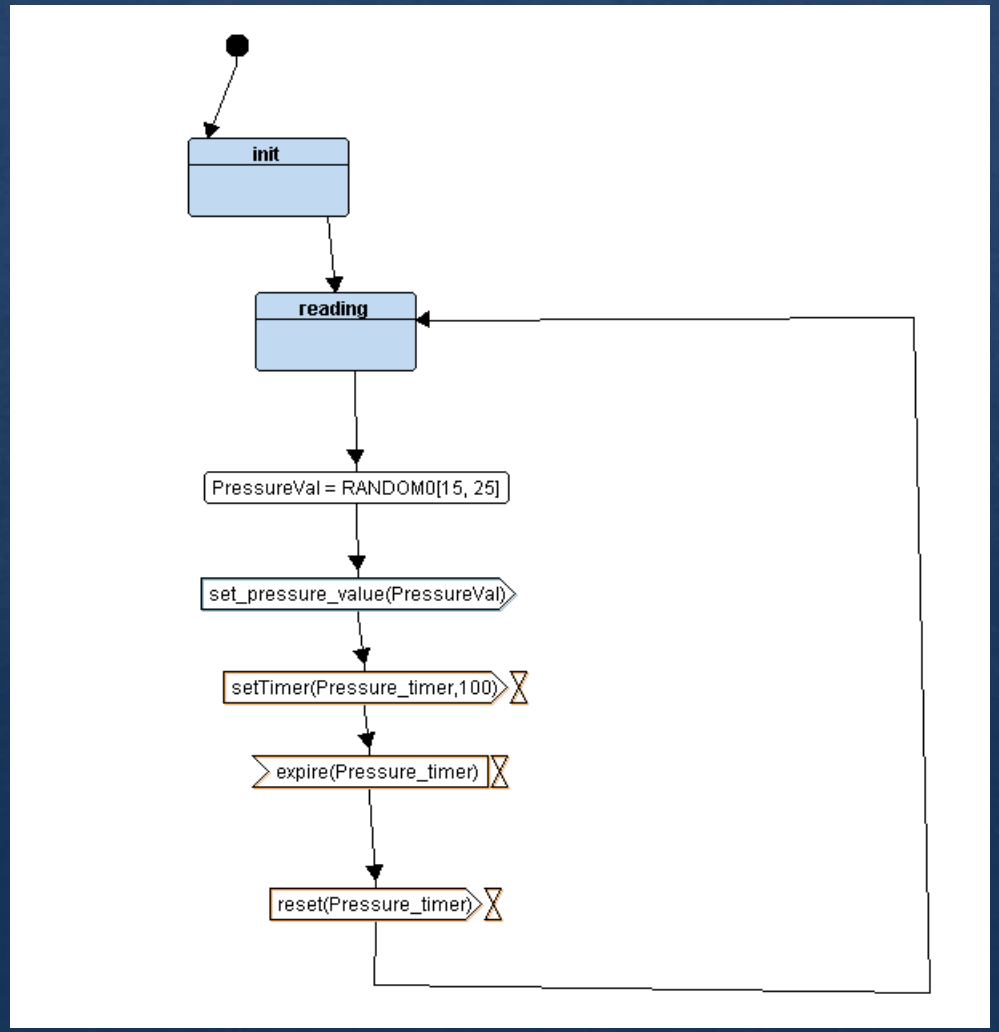
One state machine diagram per block

Block Definition Diagram and Internal Block Diagram



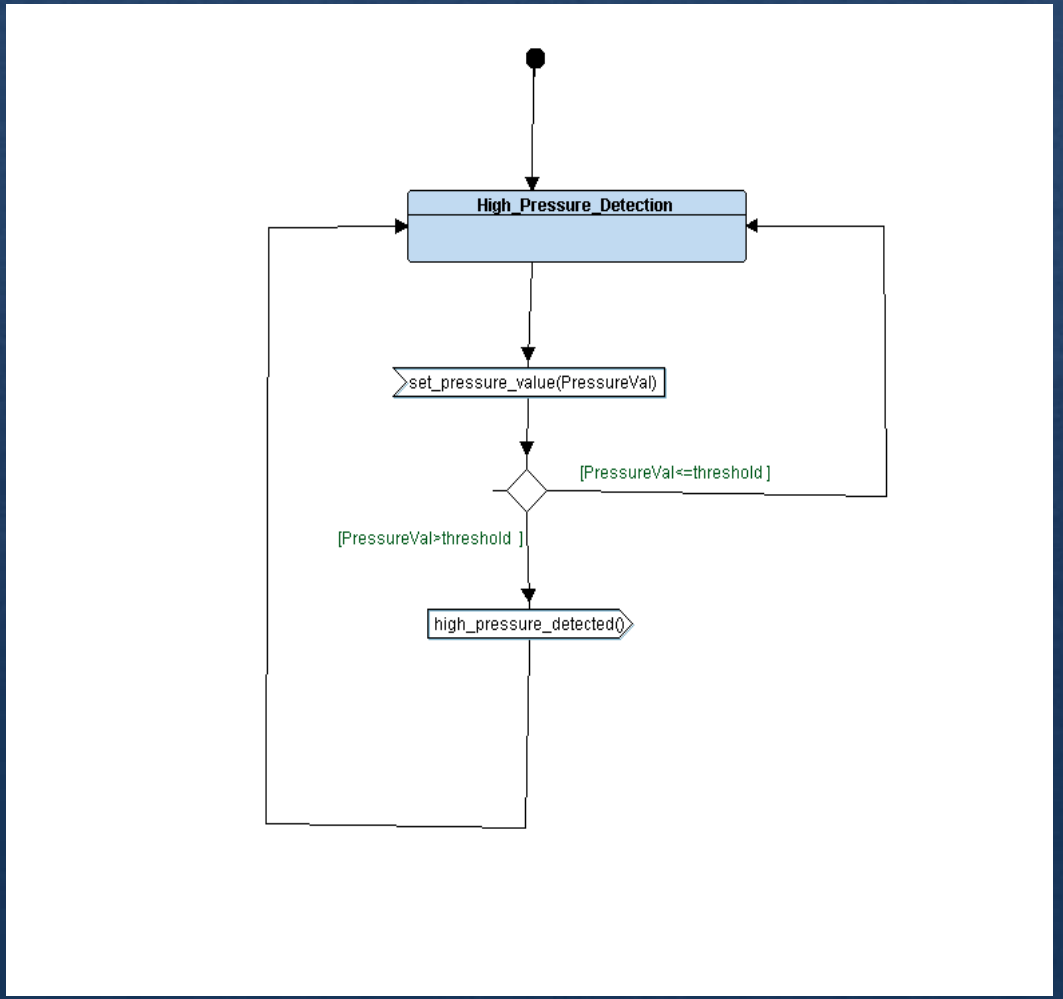
State Machine Diagram of Pressure Sensor

- Content : 2 states ,1 timer
- Take the value and send it to main algorithm block
- timer to take time until finished the process



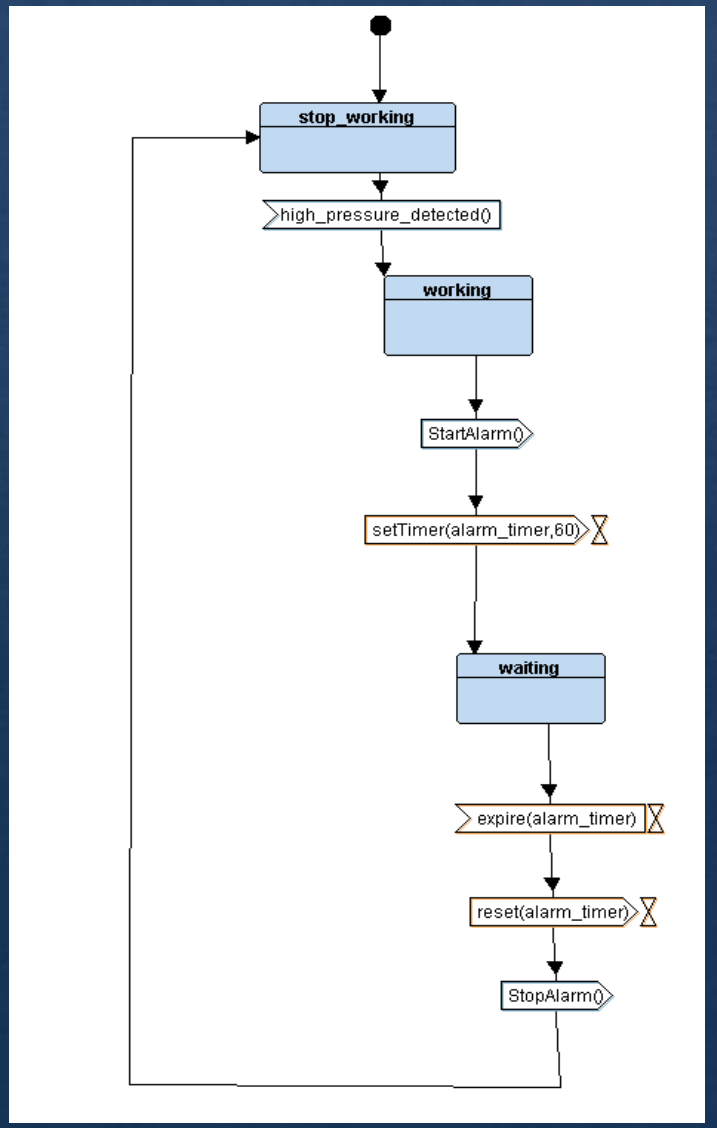
State Machine Diagram of Main Algorithm

- Content : 1 states
- Receive the value
- if pressure>threshold ,send message to Monitor
- Else return to the same state



State Machine Diagram of Alarm Monitor

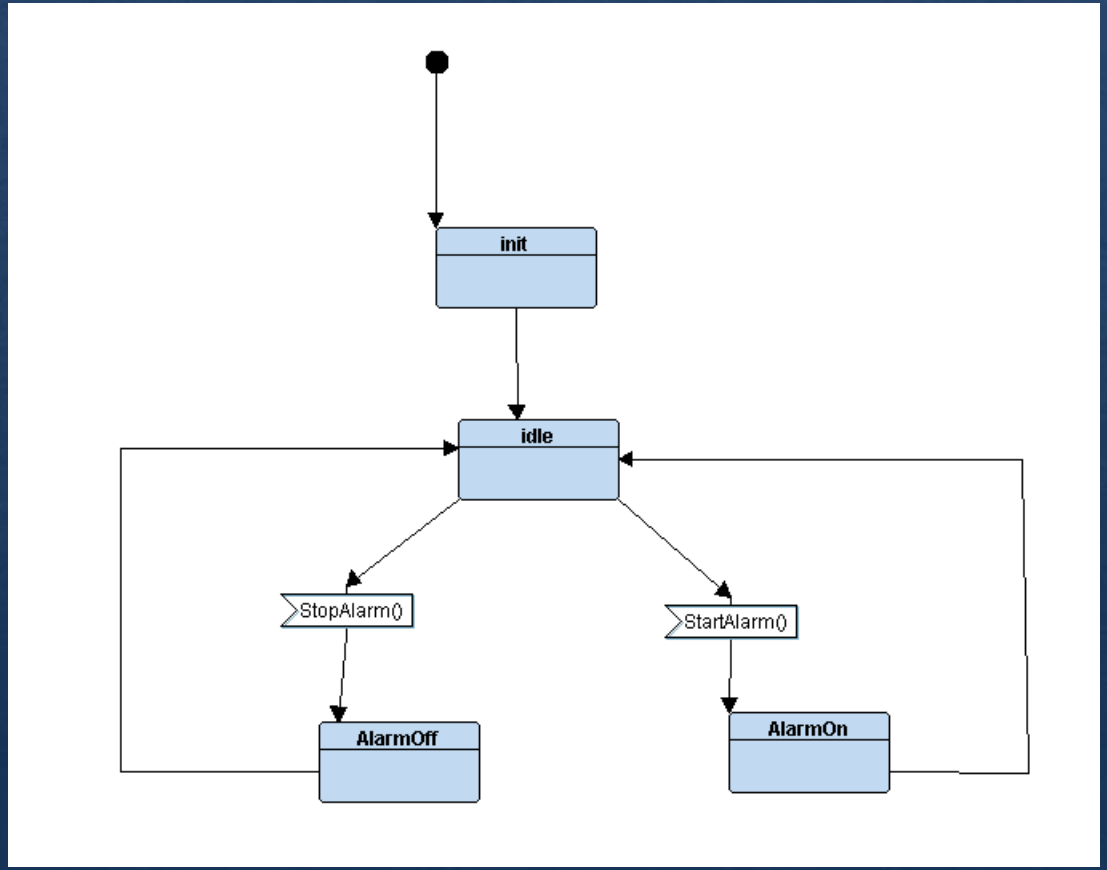
- Content : 3 states ,1 timer
- Take the Message of high pressure from Main Algorithm Block
- Working state send message to turn on alarm to actuator
- timer to 60s and send message to turn off alarm to actuator

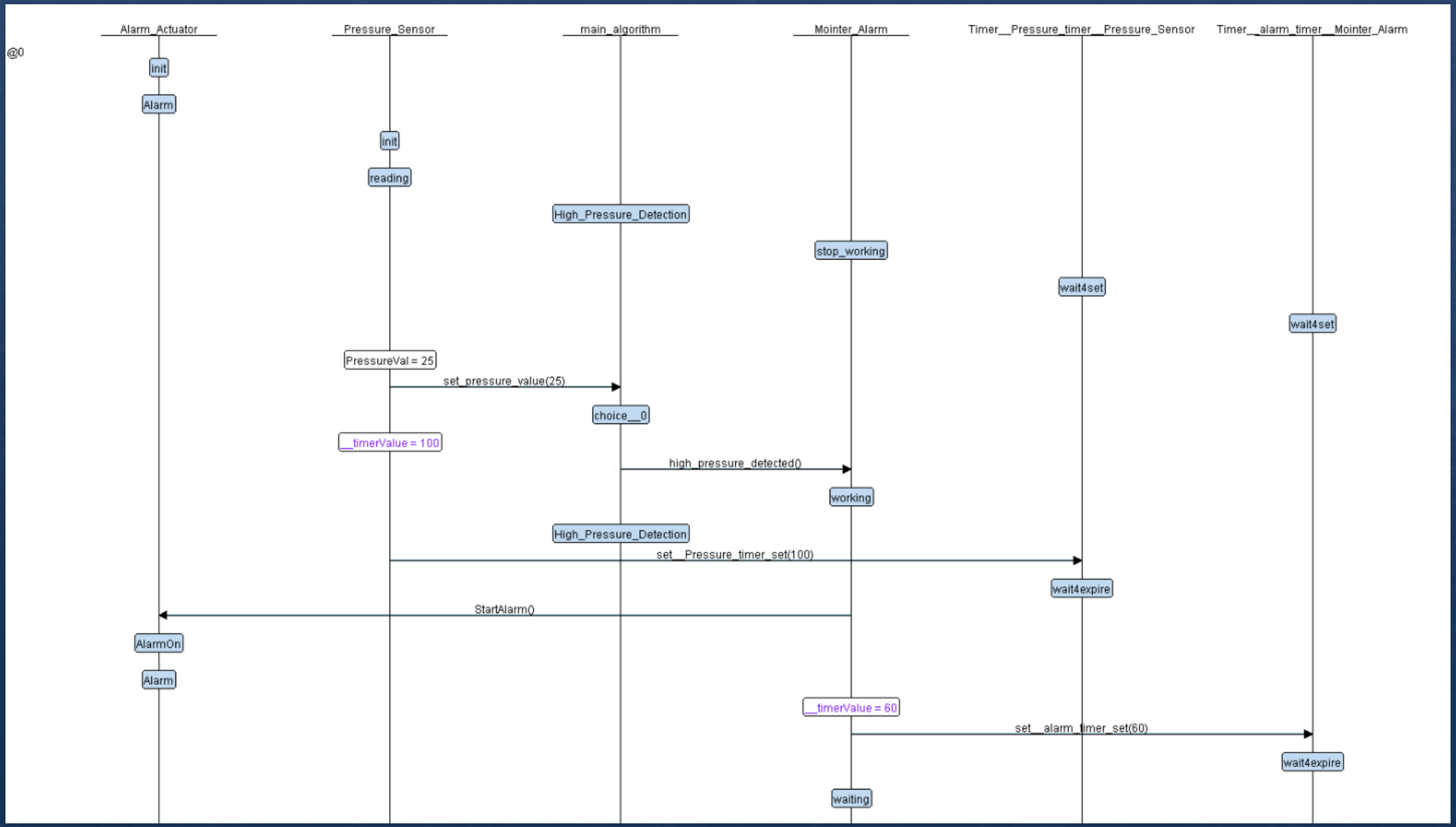


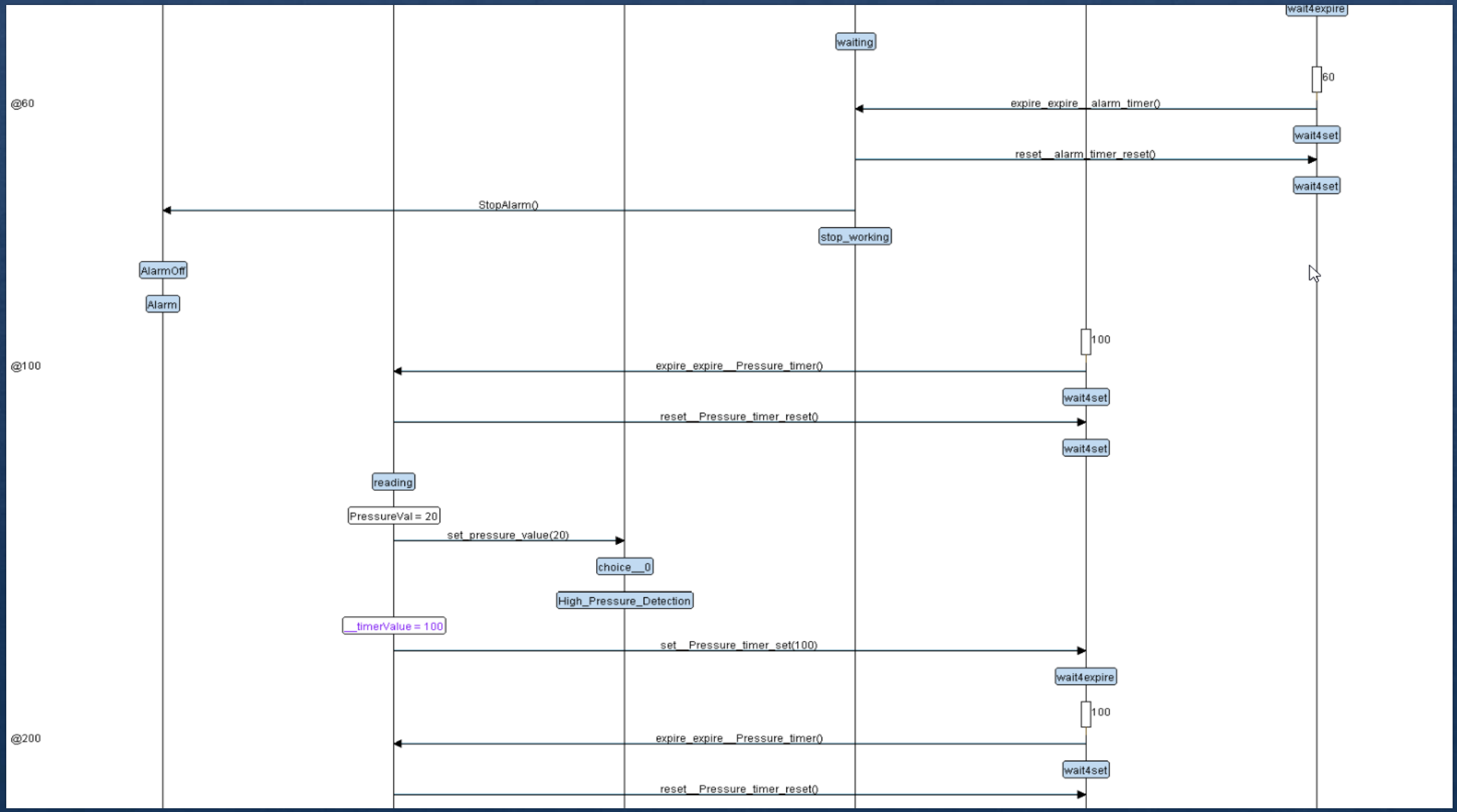
State Machine Diagram of Alarm Actuator

- Content :4 states
- Take the message from monitor TO

  Start or Stop alarm




Alarm_Actuator
Pressure_Sensor
main_algorithm
Mointer_Alarm
Timer__Pressure_timer__Pressure_Sensor
Timer__alarm_timer__Mointer_Alarm
@0
init
Alarm
init
reading
High_Pressure_Detection
stop_working
wait4set
wait4set
PressureVal = 25
set_pressure_value(25)
choice__0
__timerValue = 100
high_pressure_detected()
working
High_Pressure_Detection
set__Pressure_timer_set(100)
wait4expire
StartAlarm()
AlarmOn
Alarm
__timerValue = 60
set__alarm_timer_set(60)
wait4expire
waiting


wait4expire
waiting
60
@60
expire_expire__alarm_timer()
wait4set
reset__alarm_timer_reset()
wait4set
StopAlarm()
stop_working
AlarmOff
Alarm
100
@100
expire_expire__Pressure_timer()
wait4set
reset__Pressure_timer_reset()
wait4set
reading
PressureVal = 20
set_pressure_value(20)
choice__0
High_Pressure_Detection
__timerValue = 100
set__Pressure_timer_set(100)
wait4expire
100
@200
expire_expire__Pressure_timer()
wait4set
reset__Pressure_timer_reset()

IMPLEMENTATION

- PressueSensor.c
- PressueSensor.h
- Algorithm.c
- Algorithm.h
- AlarmMonitor.c
- AlarmMonitor.h
- Alarm_Led.c
- Alarm_Led.h
- driver.c
- driver.h
- States.h
- linker.ld
- Startup.c
- makefile

States.h

```c
#ifndef _STATES_H
#define _STATES_H
//----------------------------
//states
typedef enum
{
    //states of pressure detection
    init_PS,
    reading_PS,
    //state of algorithm
    high_presseure_detection,
    //states of monitor
    AlarmON,
    AlarmOff,
    waiting,
    //states of alarmDriver
    init_Alarm,
    idle,
    LedOn,
    LedOff
}state_t;
//----------------------------
//state functions (macros)
#define state_fun(_type_)    void ST_##_type_()
#define state(_type_)        ST_##_type_
#define state_call(_type_)   ST_##_type_()
//----------------------------
//function of transifer signals
void set_pressure_val(int pressure_val);
void high_pressure_detection();
void start_alarm();
void stop_alarm();
//----------------------------
#endif
```

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)


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

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;
}
```

PressueSensor.h

```h

#ifndef _PRESSURE_SENSOR_
#define _PRESSURE_SENSOR_

#include "states.h"
#include "driver.h"
//------------------------------
//funtions
state_fun(init_PS);
state_fun(reading_PS);
//------------------------------
//Pointer To Function
void (*ptr_PS)();
//------------------------------
#endif
```

PressueSensor.c

```c
//------------------------------
#include "PressureSensor.h"
//------------------------------

//variables
volatile unsigned int Pressure_Val=0;
state_t state_PS=reading_PS;

//------------------------------

state_fun(init_PS){
    //set the value of pointer
    ptr_PS=state(reading_PS);
}
//------------------------------

state_fun(reading_PS){
    //state name
    state_PS=reading_PS;
    //state action--->get the val of pressure
    Pressure_Val=getPressureVal();
    //state event
    set_pressure_val(Pressure_Val);
    Delay(66666);
}
```

Algorithm.h

```h
#ifndef _ALGORITHM_
#define _ALGORITHM_

#include "states.h"
#include "driver.h"
//-----------------------------
//funtions
state_fun(high_presseure_detectic
//-----------------------------
//Pointer To Function
void (*ptr_Detection)();
//-----------------------------
#endif
```

Algorithm.c

```c
//-----------------------------
#include "Algorithm.h"
#include "PressureSensor.h"
//-----------------------------

//variables
volatile unsigned int Pressure=0;
volatile unsigned int Threshold=20;
state_t state_alg=high_presseure_detection;

//-----------------------------

void set_pressure_val(int Pressure_Val){
    //recieve pressure value
    Pressure=Pressure_Val;
    //set the value of pointer
    ptr_Detection=state(high_presseure_detection);
}
//-----------------------------

state_fun(high_presseure_detection){
    //state name
    state_alg=high_presseure_detection;
    //check the high pressure
    if(Pressure>Threshold)
        high_pressure_detection();
    else
        ptr_Detection=state(high_presseure_detection);
}
```

AlarmMonitor.h

```h
#ifndef _ALARM_MONITOR_
#define _ALARM_MONITOR_

#include "states.h"
#include "driver.h"
//-----------------------------
//funtions
state_fun(AlarmON);
state_fun(AlarmOff);
state_fun(waiting);
//-----------------------------
//Pointer To Function
void (*ptr_Monitor)();
//-----------------------------

#endif
```

AlarmMonitor.c

```c
//-----------------------------
#include "Algorithm.h"
#include "AlarmMonitor.h"
//-----------------------------

//variables
state_t state_monitor=AlarmOff;

//-----------------------------

void high_pressure_detection(){
    //set the value of pointer
    ptr_Monitor=state(AlarmON);
}
//-----------------------------

state_fun(AlarmON){
    //state name
    state_monitor=AlarmON;
    //state action
    start_alarm();
    state_call(waiting);//call function
    stop_alarm();
    //return to idle
    ptr_Monitor=state(AlarmOff);
}

//-----------------------------

state_fun(AlarmOff){
    //state name
    state_monitor=AlarmOff;
    ptr_Monitor=state(AlarmOff);
}
state_fun(waiting){
    //wait 60s
    Delay(66666);
}
```

Alarm_Led.h

```h
#ifndef _ALARM_LED_
#define _ALARM_LED_

#include "states.h"
#include "driver.h"
//----------------------------
//funtions
state_fun(init_Alarm);
state_fun(LedOn);
state_fun(LedOff);
state_fun(idle);
//----------------------------
//Pointer To Function
void (*ptr_AlarmLed)();
//----------------------------
#endif
```

Alarm_Led.c

```c
#include "Alarm_Led.h"
#include "AlarmMonitor.h"
//variables
state_t state_Alarm=LedOn;
//----------------------------
state_fun(init_Alarm){
    //set the value of pointer
    ptr_AlarmLed=state(idle);
}
//----------------------------
state_fun(idle){
    //return here
}
//----------------------------
void start_alarm(){
    //set the value of pointer
    state_call(LedOn);//call
}
//----------------------------
void stop_alarm(){
    //set the value of pointer
    state_call(LedOff);//call
}
//----------------------------
state_fun(LedOn){
    //state name
    state_Alarm=LedOn;
    //state action
    Set_Alarm_actuator(0);
    ptr_AlarmLed=state(idle);
}
//----------------------------
state_fun(LedOff){
    //state name
    state_Alarm=LedOff;
    //state action
    Set_Alarm_actuator(1);
    delay(10000);
    ptr_AlarmLed=state(idle);
}
```

linker.ld

```
MEMORY {
    SRAM(RWX) : ORIGIN =0x20000000 , LENGTH =20K
    FLASH(RX) : ORIGIN =0x08000000 , LENGTH =128K
}

SECTIONS{
    .text :
    {
        *(.vectors*)
        *(.text*)
        *(.rodata*)
        . = ALIGN(4);
        _end_text = . ;
    } > FLASH
    .data :
    {
        _start_data = . ;
        *(.data*)
        . = ALIGN(4);
        _end_data = . ;
    } > SRAM AT> FLASH
    .bss :
    {
        _start_bss = . ;
        *(.bss*)
        . = ALIGN(4);
        _end_bss = . ;
    } > SRAM
}
```

makefile

```
CC = arm-none-eabi-
CFLAGS = -gdwarf-2 -mcpu=cortex-m4 -mthumb
INGS = -I.

#wildcard
SRC =$(wildcard *.c)
AS =$(wildcard *.s)
#command to all object files
OBJ =$(SRC:.c=.o)
ASOBJ =$(AS:.s=.o)

#project name
project_name = High_Pressure_Detection

#buliding
build : $(project_name).bin
    echo DONE...

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

%.o : %.s
    $(CC)AS $< $(CFLAGS) $(INGS) -std=c99 -o $@

$(project_name).elf : $(ASOBJ) $(OBJ)
    $(CC)ld -T linker.ld $(ASOBJ) $(OBJ) -o $@ -Map Map_file.map
    cp $(project_name).elf  $(project_name).hex

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

#clean all files
cleanAll:
    rm -f  $(ASOBJ) $(OBJ) $(project_name).bin $(project_name).elf $(project_name).hex
#clean after the object files
clean:
    rm -f  $(project_name).bin $(project_name).elf $(project_name).hex
```

Startup.c

```c
volatile unsigned char _end_text;
volatile unsigned char _start_data;
volatile unsigned char _end_data;
volatile unsigned char _start_bss;
volatile unsigned char _end_bss;
//init handlers
void Default_handler();
void reset_handler();
void NMI_handler()  __attribute__((weak,alias("Default_handler")));
//init satckTop
static volatile unsigned int stackTop[256];
//init array of Vector Section
void (*const arr[])() __attribute__((section(".vectors")))={
    (void(*)())  stackTop+(sizeof(stackTop)),
    &reset_handler,
    &NMI_handler
};
//reset
void reset_handler(){
    unsigned int i;
    //copy .data
    volatile unsigned char data_size =(volatile unsigned char)&_end_data - (volatile unsigned char)&_start_data;
    volatile unsigned char * ptr1 =&_end_text ;
    volatile unsigned char * ptr2 =&_start_data ;
    for(i=0;i<data_size;i++){
        * ptr2 =* ptr1;
        ptr1++;ptr2++;
    }
    //init .bss
    volatile unsigned char bss_size =(volatile unsigned char)&_end_bss - (volatile unsigned char)&_start_bss;
    ptr1 =&_start_bss ;
    for(i=0;i<data_size;i++){
        * ptr1 =(volatile unsigned char)0;
        ptr1++;
    }
    main();
}
void Default_handler(){
    reset_handler();
}
```

Mapfile.map

```
Memory Configuration

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

```
.data           0x20000000       0x10 load address 0x08000460
                0x20000000                _start_data = .
 *(.data*)
 .data          0x20000000        0x1 Alarm_Led.o
                0x20000000                state_Alarm
 .data          0x20000001        0x1 AlarmMonitor.o
                0x20000001                state_monitor
 *fill*         0x20000002        0x2
 .data          0x20000004        0x8 Algorithm.o
                0x20000004                Threshold
                0x20000008                state_alg
 .data          0x2000000c        0x0 driver.o
 .data          0x2000000c        0x0 main.o
 .data          0x2000000c        0x1 PressueSensor.o
                0x2000000c                state_PS
 .data          0x2000000d        0x0 startup.o
                0x20000010                . = ALIGN (0x4)
 *fill*         0x2000000d        0x3
                0x20000010                _end_data = .

.igot.plt       0x20000010        0x0 load address 0x08000470
 .igot.plt      0x00000000        0x0 Alarm_Led.o

.bss            0x20000010      0x41d load address 0x08000470
                0x20000010                _start_bss = .
 *(.bss*)
 .bss           0x20000010        0x0 Alarm_Led.o
 .bss           0x20000010        0x0 AlarmMonitor.o
 .bss           0x20000010        0x4 Algorithm.o
                0x20000010                Pressure
 .bss           0x20000014        0x0 driver.o
 .bss           0x20000014        0x0 main.o
 .bss           0x20000014        0x4 PressueSensor.o
                0x20000014                Pressure_Val
 .bss           0x20000018      0x400 startup.o
                0x20000418                . = ALIGN (0x4)
                0x20000418                _end_bss = .
```

```
.text           0x08000000      0x460
 *(.vectors*)
 .vectors       0x08000000        0xc startup.o
                0x08000000                arr
 *(.text*)
 .text          0x0800000c       0xa0 Alarm_Led.o
                0x0800000c                ST_init_Alarm
                0x08000028                ST_idle
                0x08000034                start_alarm
                0x08000040                stop_alarm
                0x0800004c                ST_LedOn
                0x0800007c                ST_LedOff
 .text          0x080000ac       0x90 AlarmMonitor.o
                0x080000ac                high_pressure_detection
                0x080000c8                ST_AlarmON
                0x080000fc                ST_AlarmOff
                0x08000128                ST_waiting
 .text          0x0800013c       0x74 Algorithm.o
                0x0800013c                set_pressure_val
                0x0800016c                ST_high_presseure_detection
 .text          0x080001b0      0x10c driver.o
                0x080001b0                Delay
                0x080001d4                getPressureVal
                0x080001ec                Set_Alarm_actuator
                0x0800023c                GPIO_INITIALIZATION
 .text          0x080002bc       0x70 main.o
                0x080002bc                setup
                0x080002fc                main
 .text          0x0800032c       0x60 PressueSensor.o
                0x0800032c                ST_init_PS
                0x08000348                ST_reading_PS
 .text          0x0800038c       0xd4 startup.o
                0x0800038c                reset_handler
                0x08000454                Default_handler
                0x08000454                NMI_handler
 *(.rodata*)
                0x08000460                . = ALIGN (0x4)
                0x08000460                _end_text = .
```

Building with makefile in terminal

```
Office@DESKTOP-AF0NRE0 MINGW32 /e/github/Unit5_projects/High_Pressure_Detection/
Code
$ make
arm-none-eabi-gcc -c Alarm_Led.c -gdwarf-2 -mcpu=cortex-m4 -mthumb -I. -o Alarm_Led.o
arm-none-eabi-gcc -c AlarmMonitor.c -gdwarf-2 -mcpu=cortex-m4 -mthumb -I. -o AlarmMonitor.o
arm-none-eabi-gcc -c Algorithm.c -gdwarf-2 -mcpu=cortex-m4 -mthumb -I. -o Algorithm.o
arm-none-eabi-gcc -c driver.c -gdwarf-2 -mcpu=cortex-m4 -mthumb -I. -o driver.o
arm-none-eabi-gcc -c main.c -gdwarf-2 -mcpu=cortex-m4 -mthumb -I. -o main.o
arm-none-eabi-gcc -c PressueSensor.c -gdwarf-2 -mcpu=cortex-m4 -mthumb -I. -o PressueSensor.o
arm-none-eabi-gcc -c startup.c -gdwarf-2 -mcpu=cortex-m4 -mthumb -I. -o startup.o
startup.c: In function 'reset_handler':
startup.c:22:36: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
startup.c:22:73: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
startup.c:30:35: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
startup.c:30:71: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
arm-none-eabi-ld -T linker.ld  Alarm_Led.o AlarmMonitor.o Algorithm.o driver.o main.o PressueSensor.o startup.o -o High_Pressure_Detection.elf -Map Map_file.map
cp High_Pressure_Detection.elf  High_Pressure_Detection.hex
arm-none-eabi-objcopy -O binary High_Pressure_Detection.elf High_Pressure_Detection.bin
echo DONE...
DONE...
```

Sections of project.elf

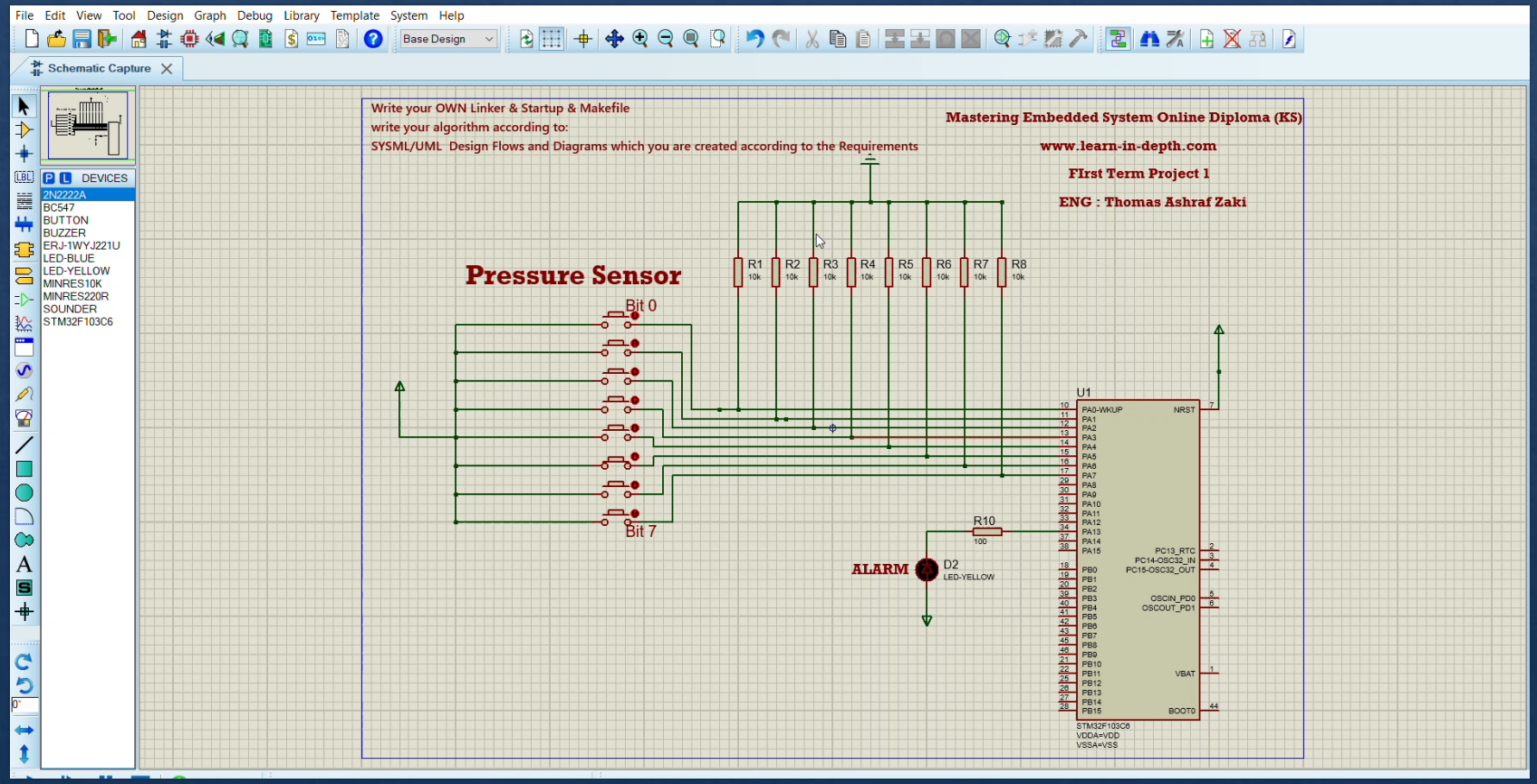
```
$ arm-none-eabi-objdump.exe -h High_Pressure_Detection.elf

High_Pressure_Detection.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000460  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000010  20000000  08000460  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          0000041d  20000010  08000470  00010010  2**2
                  ALLOC
  3 .debug_info   0000091c  00000000  00000000  00010010  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 00000496  00000000  00000000  0001092c  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000003f8  00000000  00000000  00010dc2  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000e0  00000000  00000000  000111ba  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000370  00000000  00000000  0001129a  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    000002fe  00000000  00000000  0001160a  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  00011908  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00011919  2**0
                  CONTENTS, READONLY
 11 .debug_frame  000002c0  00000000  00000000  0001194c  2**2
                  CONTENTS, READONLY, DEBUGGING
```

Symbols of project.elf

```
$ arm-none-eabi-nm.exe High_Pressure_Detection.elf
20000418 B _end_bss
20000010 D _end_data
08000460 T _end_text
20000010 B _start_bss
20000000 D _start_data
08000000 T arr
08000454 T Default_handler
080001b0 T Delay
080001d4 T getPressureVal
0800023c T GPIO_INITIALIZATION
080000ac T high_pressure_detection
080002fc T main
08000454 W NMI_handler
20000010 B Pressure
20000014 B Pressure_Val
20000418 B ptr_AlarmLed
20000420 B ptr_Detection
2000041c B ptr_Monitor
20000424 B ptr_PS
0800038c T reset_handler
080001ec T Set_Alarm_actuator
0800013c T set_pressure_val
080002bc T setup
080000fc T ST_AlarmOff
080000c8 T ST_AlarmON
0800016c T ST_high_presseure_detection
08000028 T ST_idle
0800000c T ST_init_Alarm
0800032c T ST_init_PS
0800007c T ST_LedOff
0800004c T ST_LedOn
08000348 T ST_reading_PS
08000128 T ST_waiting
20000018 b stackTop
08000034 T start_alarm
20000000 D state_Alarm
20000008 D state_alg
20000001 D state_monitor
2000000c D state_PS
08000040 T stop_alarm
20000004 D Threshold
```

Video of Simulation

Link: click here



THANKS

- Under the supervision of
- Eng: Keroloes Shenouda Khalil (here)