2023

Ahmed Essam Elmola

MASTERING EMBEDDED SYSTEMS ONLINE DIPLOMA

ahmedelmola224@gmail.com (learn-in-depth.com)

First Term: (Final Project 1)

Case Study:

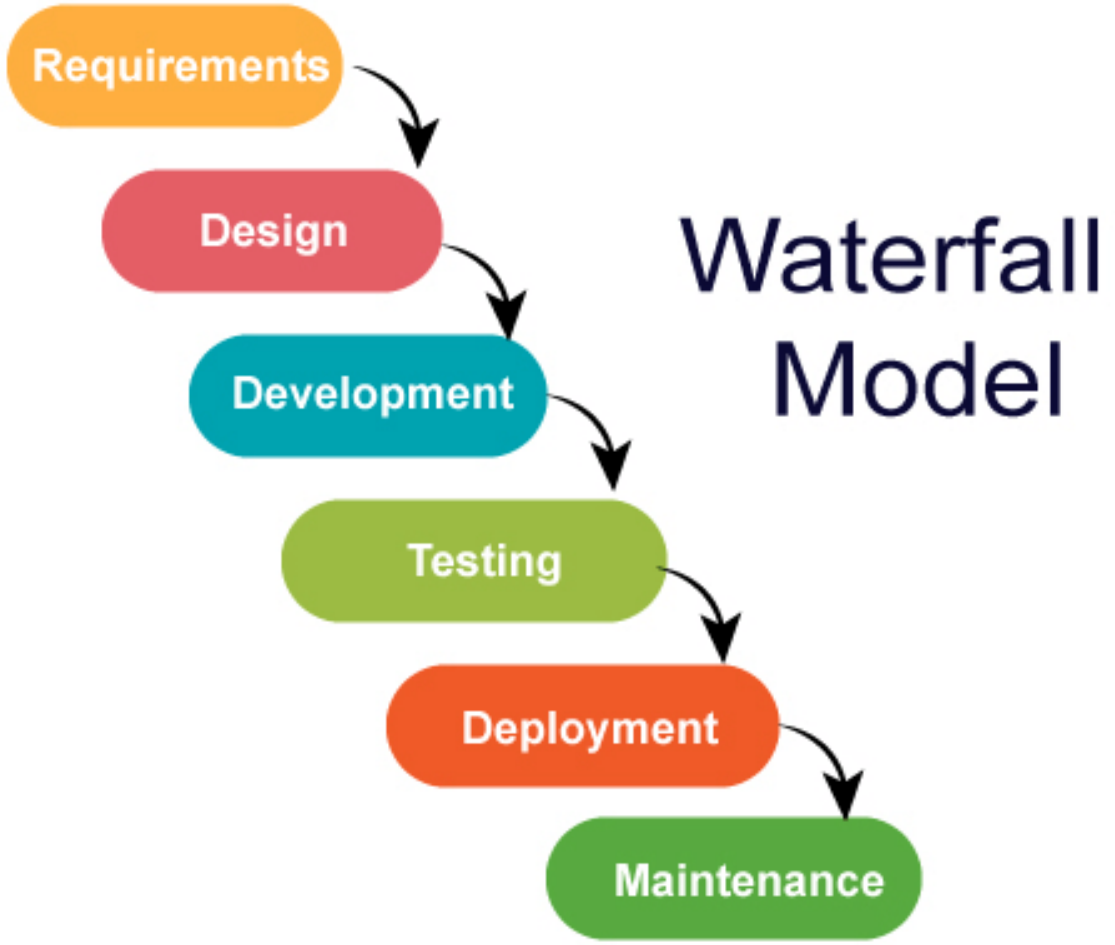
- A pressure controller informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin.
- The alarm duration equals 60 seconds.
- keeps track of the measured values(optional).

ASSUMPTIONS

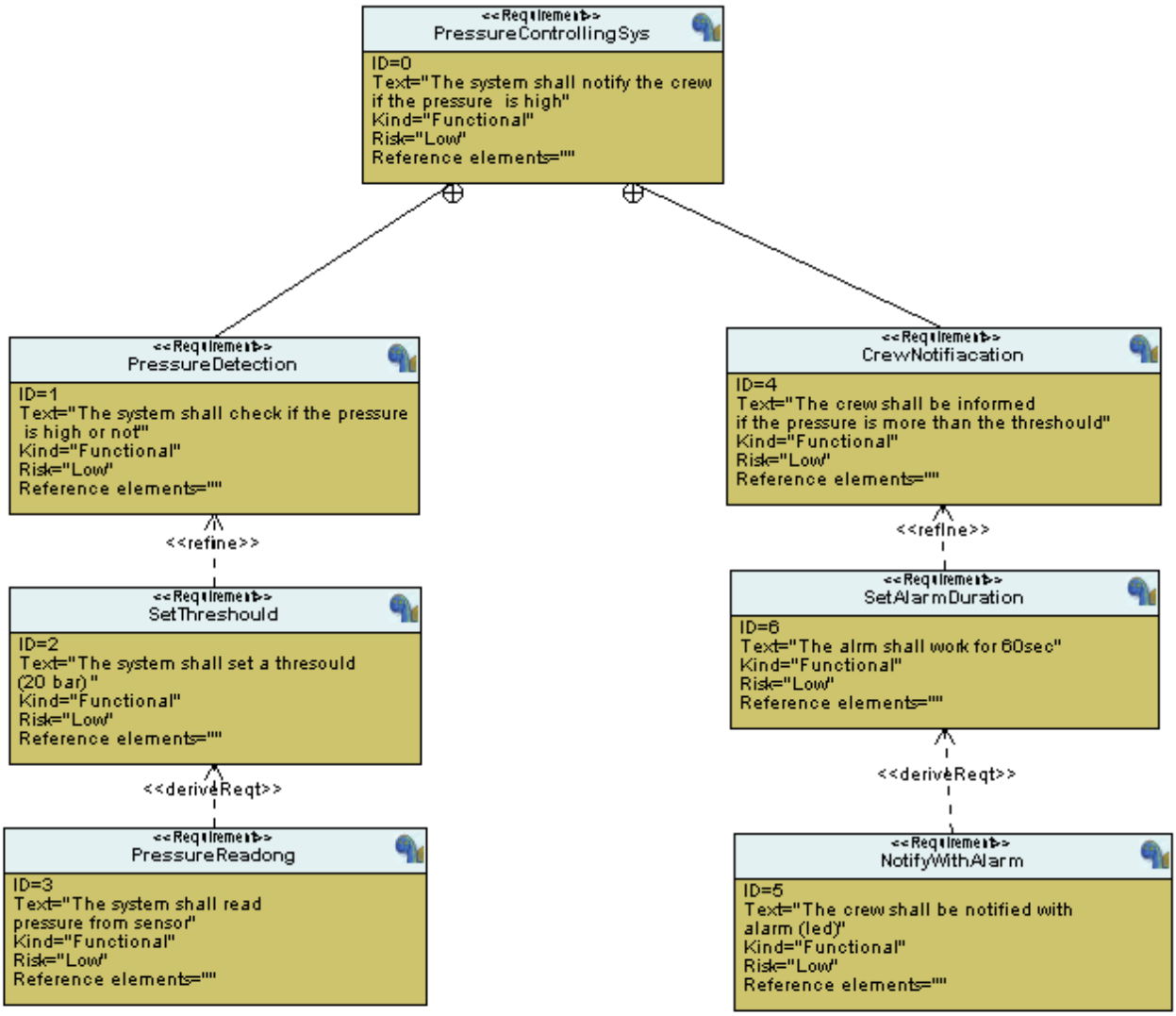
- The controller set up and shutdown procedures are not modeled.
- The controller maintenance is not modeled.
- The pressure sensor never fails.
- The alarm never fails.
- The controller never faces power cut.

Methodology:

As it’s a small-scale project, we are going to use the water full method.

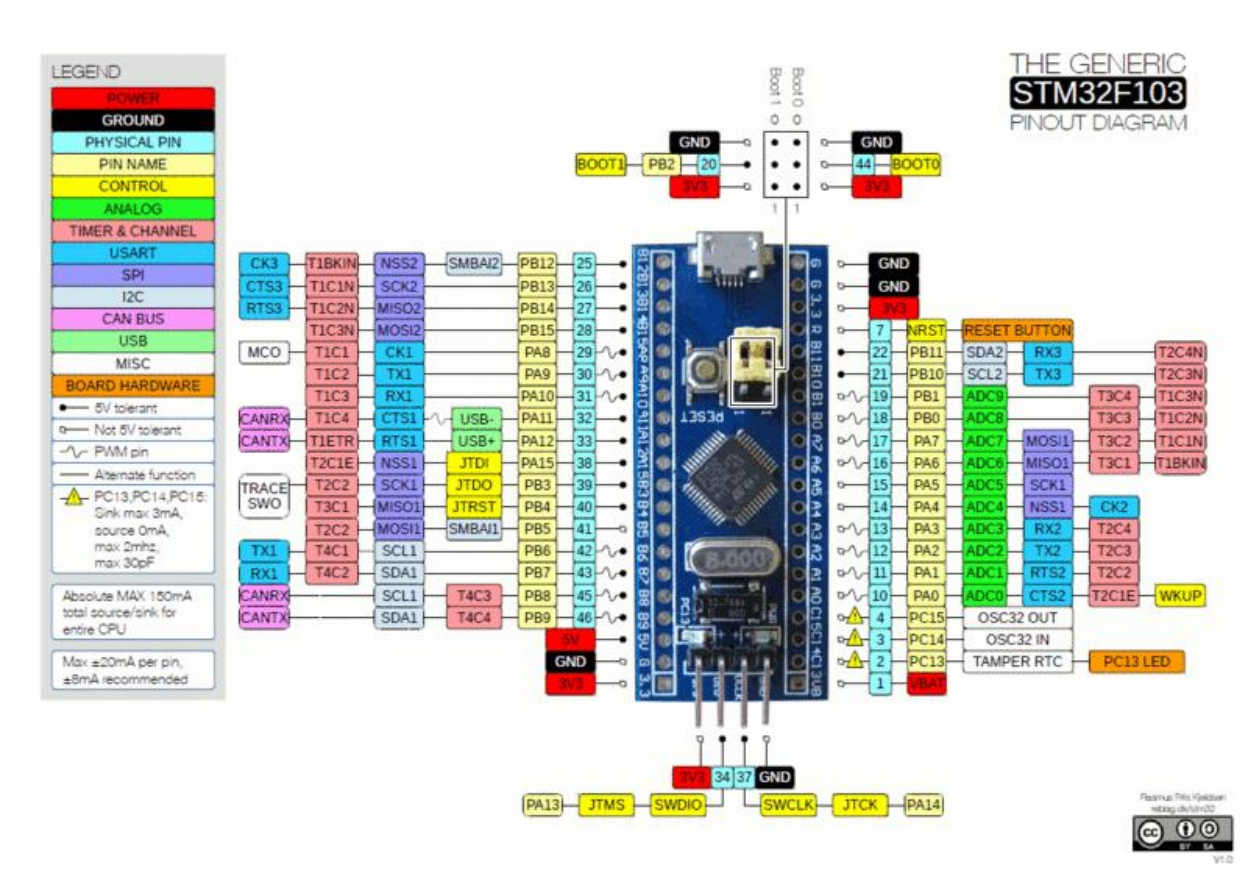


System-Requirements:



Space-Exploration:

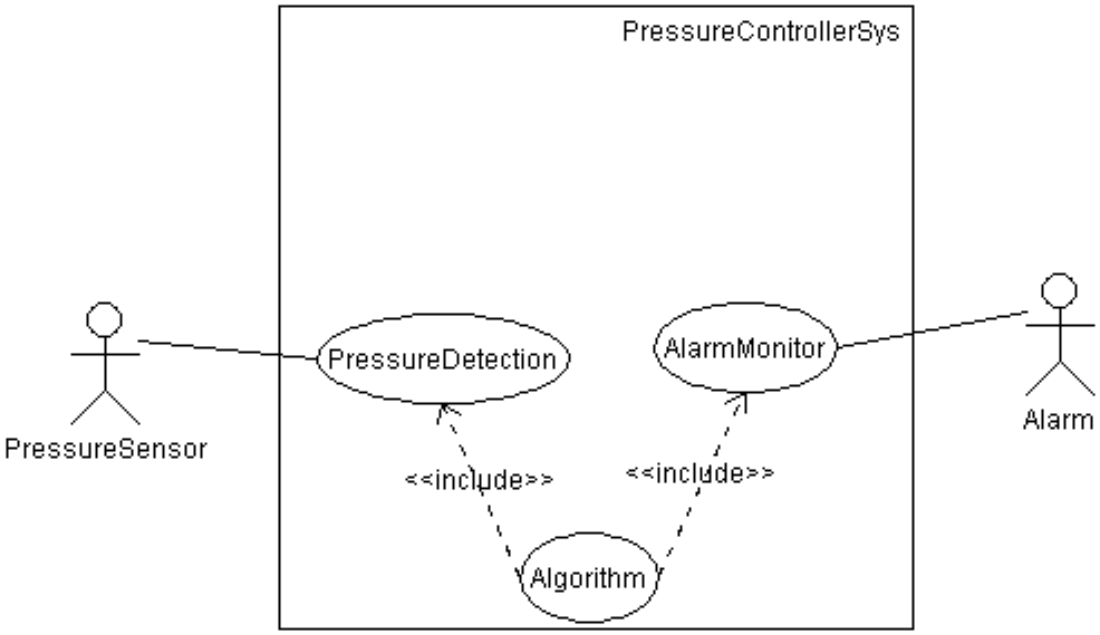
As it’s a small-scale project, we are going to use only one MCU (Stm32f103CX) with a cortex-m3 processor which is suitable for this project.



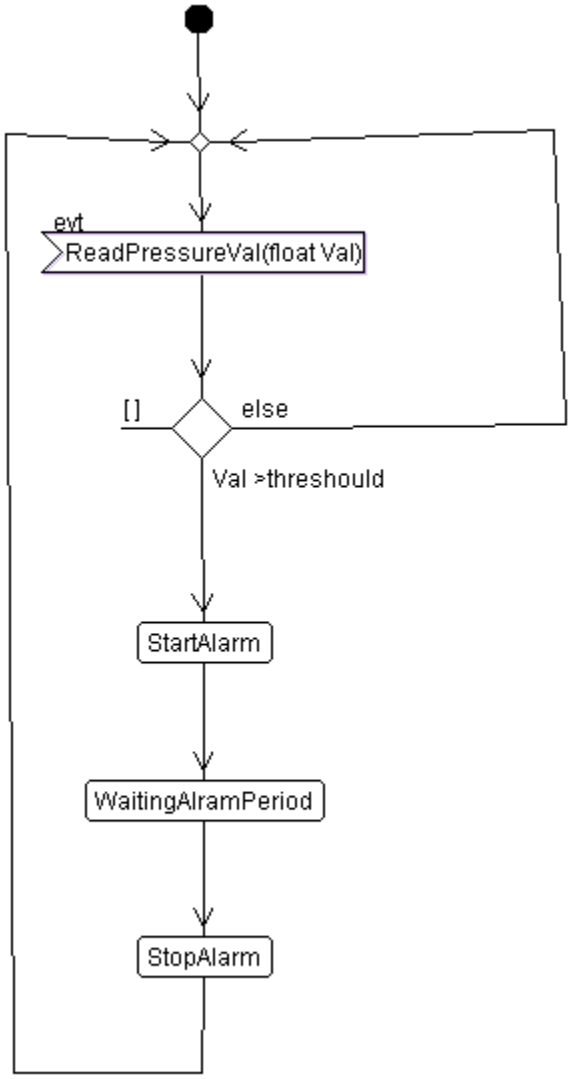
System-Analysis:

At this stage we need to understand the project very well, then discuss that with the client.

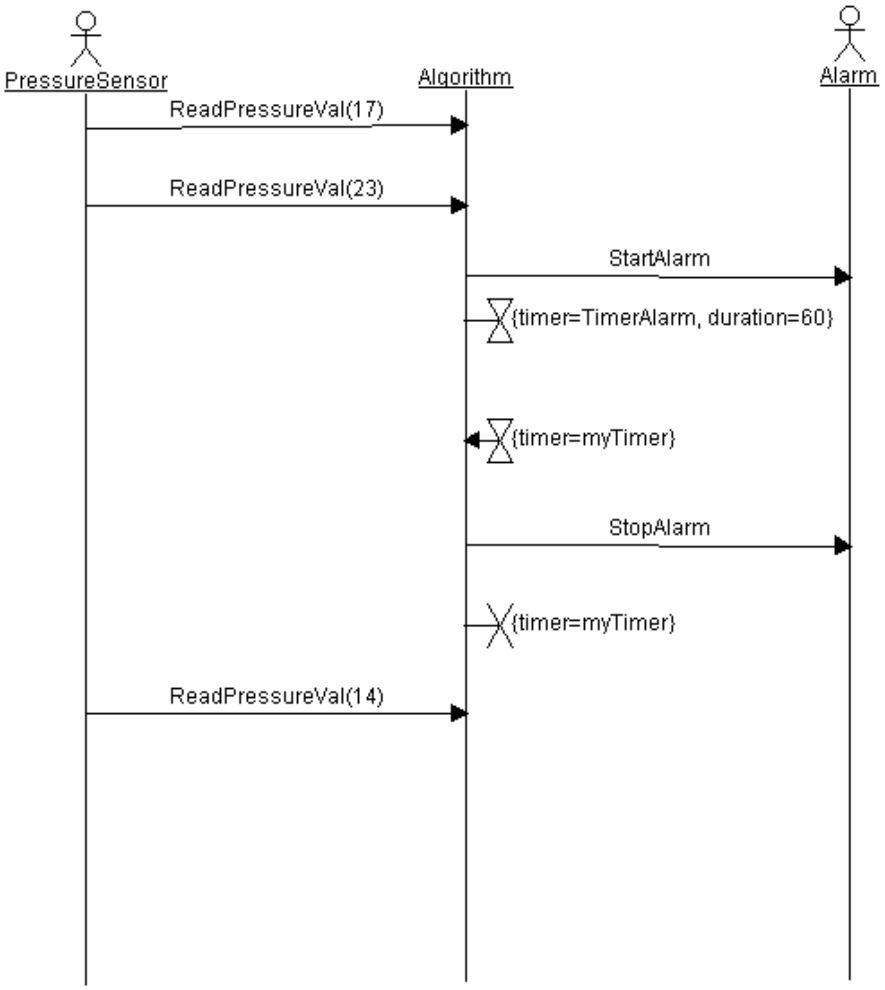
USE CASE DIAGRAM



ACTIVITY DIAGRAM

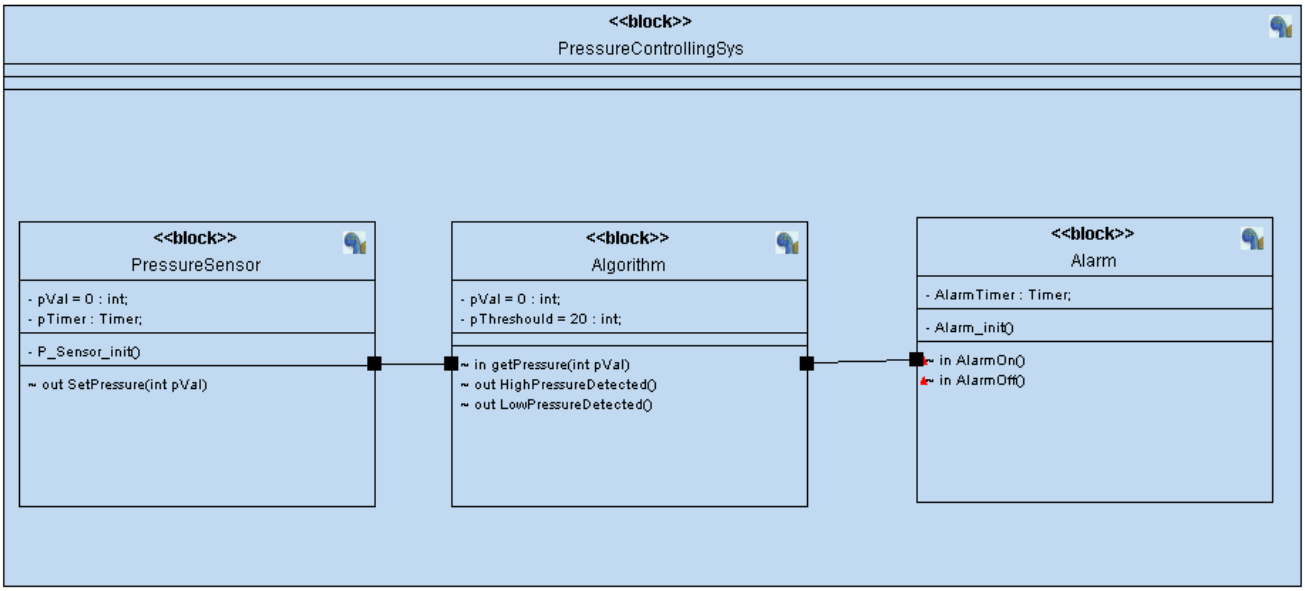


SEQUENCE DIAGRAM

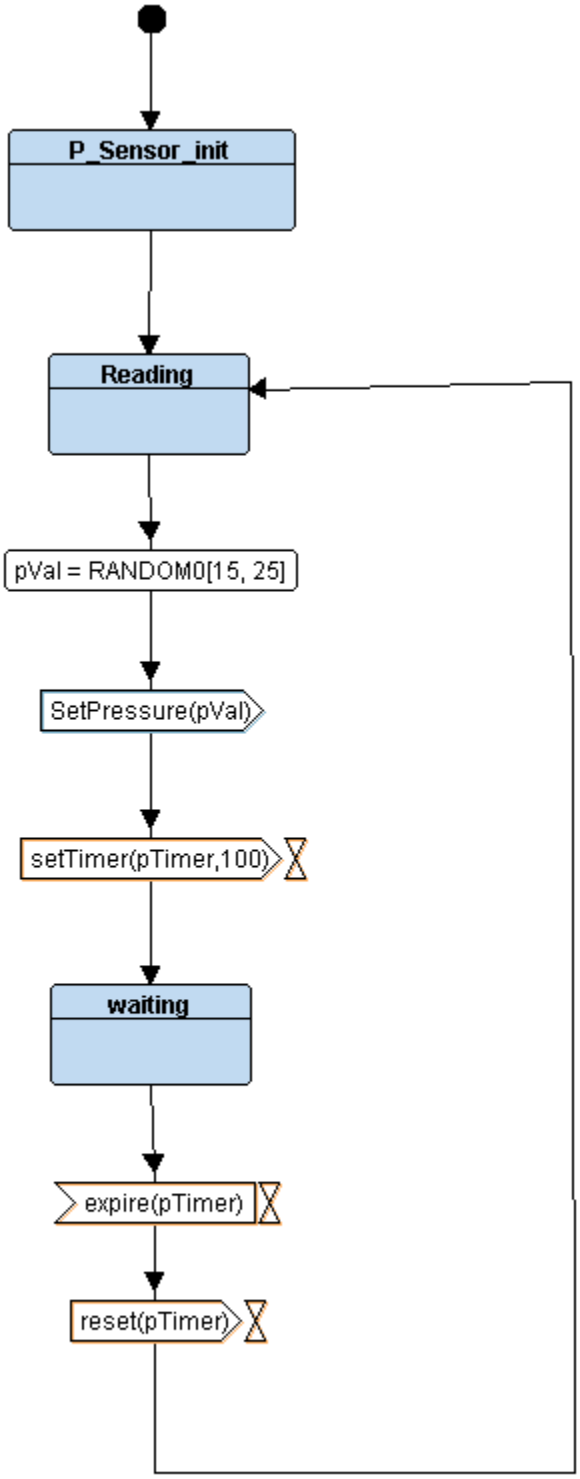


System-Design:

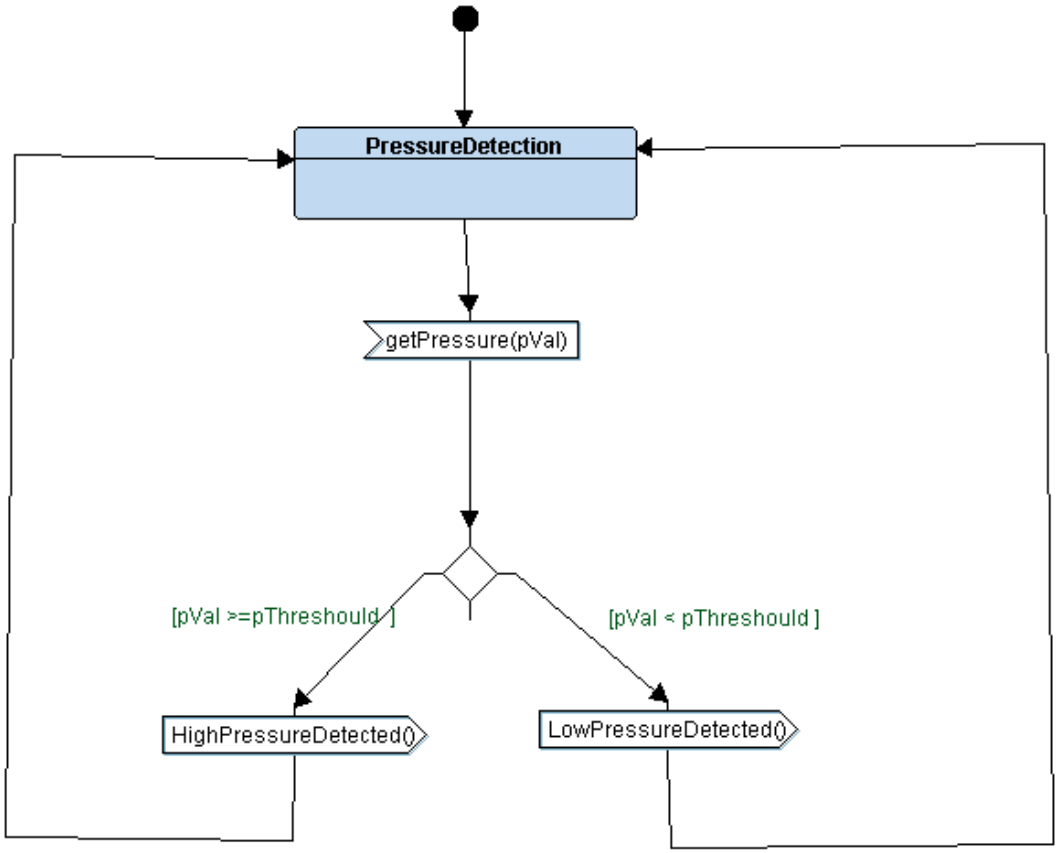
PRESSURE CONTROLLING SYS DESIGN



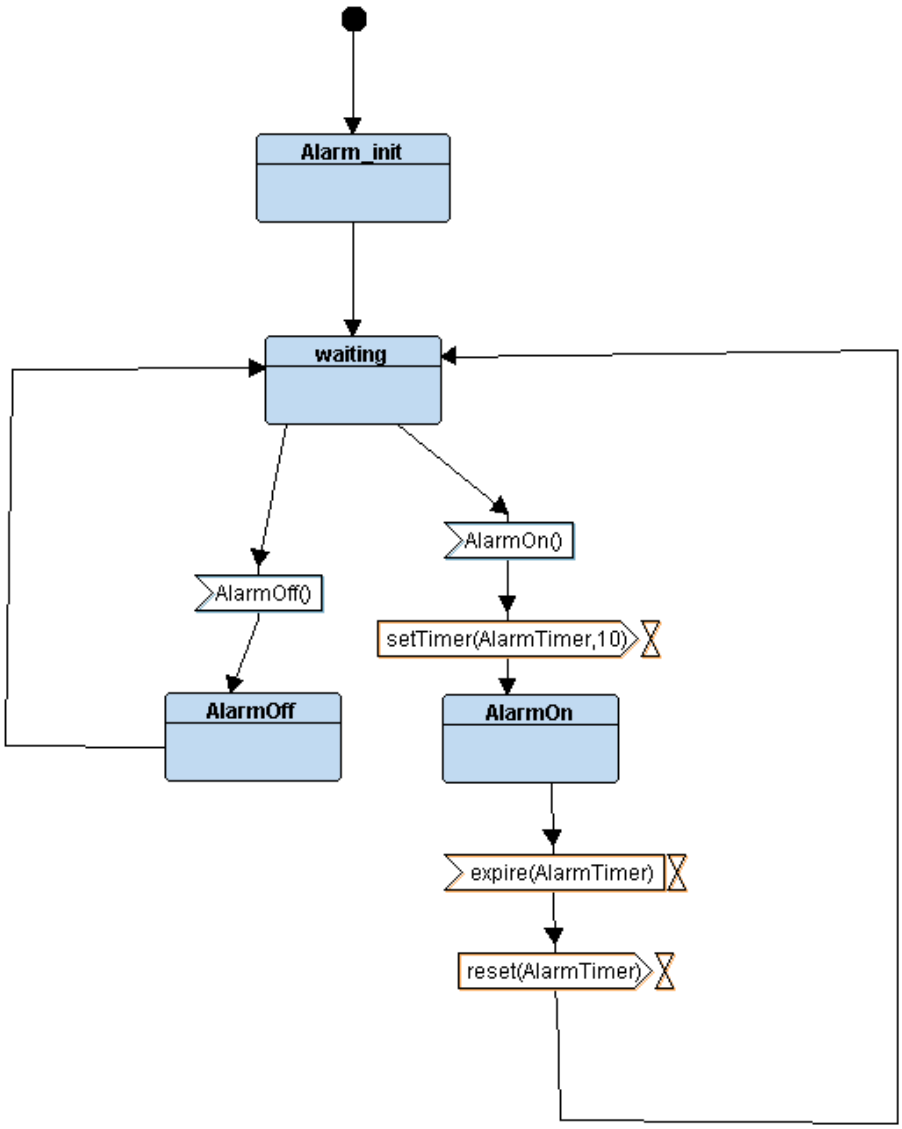
PRESSURE SENSOR STATE MACHINE DIAGRAM

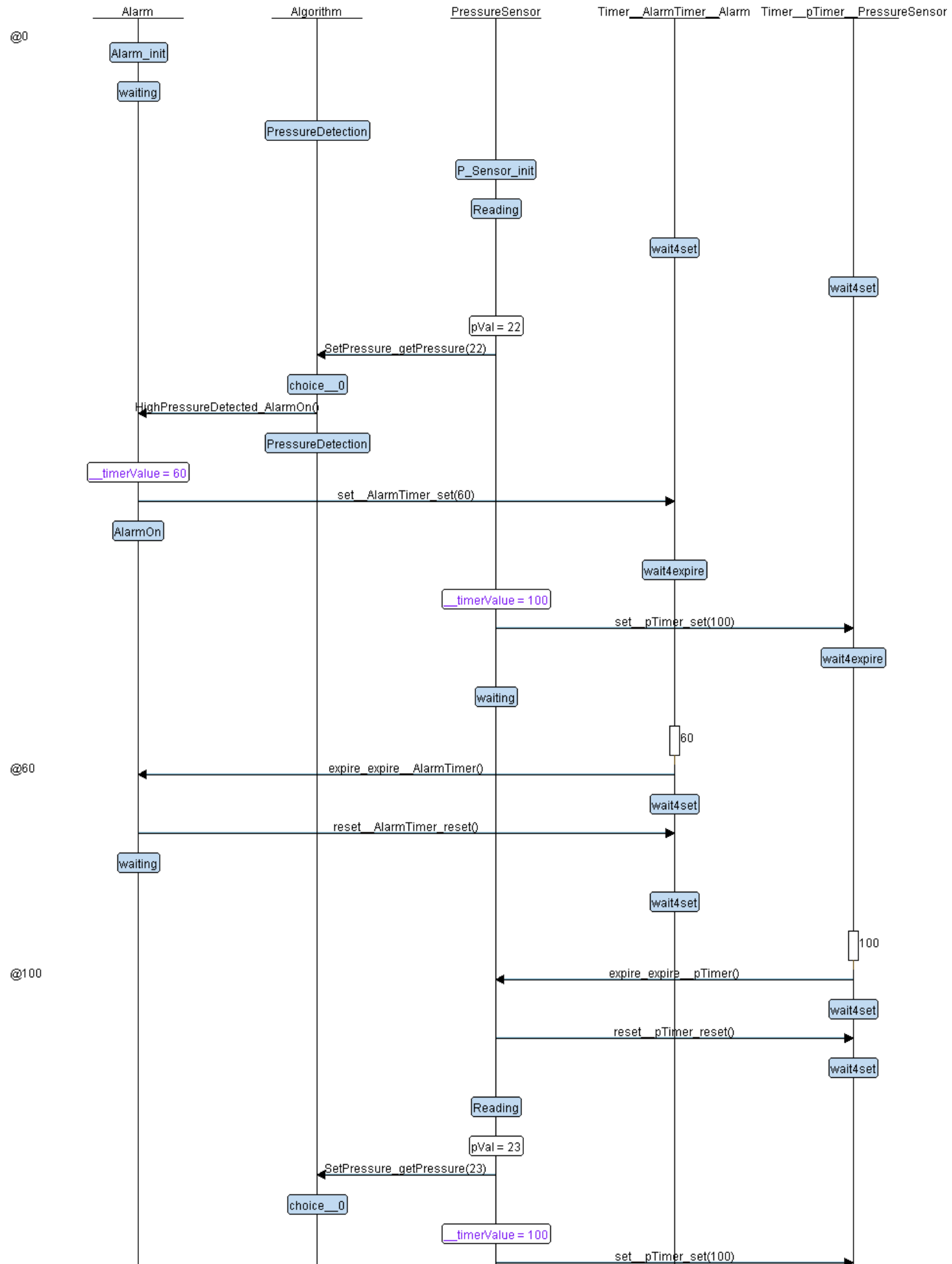


ALGORITHM STATE MACHINE DIAGRAM



ALARM STATE MACHINE DIAGRAM




Alarm
Algorithm
PressureSensor
Timer__AlarmTimer__Alarm
Timer__pTimer__PressureSensor
@0
Alarm_init
waiting
PressureDetection
P_Sensor_init
Reading
wait4set
wait4set
pVal = 22
SetPressure_getPressure(22)
choice__0
HighPressureDetected_AlarmOn()
PressureDetection
__timerValue = 60
set__AlarmTimer_set(60)
AlarmOn
wait4expire
__timerValue = 100
set__pTimer_set(100)
wait4expire
waiting
60
@60
expire_expire__AlarmTimer()
wait4set
reset__AlarmTimer_reset()
waiting
wait4set
100
@100
expire_expire__pTimer()
wait4set
reset__pTimer_reset()
wait4set
Reading
pVal = 23
SetPressure_getPressure(23)
choice__0
__timerValue = 100
set__pTimer_set(100)

SYMBOLS

```
20000020 B _E_bss
2000000c D _E_data
0800044c T _E_text
2000000c B _S_bss
20000000 D _S_data
20001020 B _stack_top
0800001c T Alarm_init
2000000c b alarm_state
20001024 B alarm_state_id
20000014 b algo_status_id
08000440 W Bus_Fault
08000440 T Default_Handler
08000194 T Delay
080001b8 T getPressureVal
08000220 T GPIO_INITIALIZATION
08000440 W H_Fault_Handler
080002a0 T main
08000440 W MM_Fault_Handler
08000440 W NMI_Handler
080002d8 T P_Sensor_init
20001020 B pf_alarm_state
20000004 D pf_algo_state
20000008 D pf_ps_state
2000001c b ps_status_id
20000000 d pThreshould
20000018 b pVal
20000010 b pVal
08000388 T Reset_Handler
080001d0 T Set_Alarm_actuator
08000104 T set_alarm_state
08000174 T set_pressure
080000d4 T ST_ALARM_OFF
0800009c T ST_ALARM_ON
08000048 T ST_ALARM_WAITING
08000124 T ST_ALGO_PREESURE_DETECTION
08000304 T ST_PS_READING
08000358 T ST_PS_WAITING
08000440 W Usage_Fault_Handler
08000000 T vectors
```

SECTIONS

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

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000044c  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         0000000c  20000000  0800044c  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001019  2000000c  08000458  0001000c  2**2
                  ALLOC
  3 .debug_info   000006b9  00000000  00000000  0001000c  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 000003fb  00000000  00000000  000106c5  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000328  00000000  00000000  00010ac0  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000c0  00000000  00000000  00010de8  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   000002b8  00000000  00000000  00010ea8  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    000002b7  00000000  00000000  00011160  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  00011417  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00011428  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000230  00000000  00000000  0001145c  2**2
                  CONTENTS, READONLY, DEBUGGING
```

STARTUP

```c
/*startup
Ahmed Essam*/
#include "Platform_Types.h"

extern void main();
void Reset_Handler();
void Default_Handler();

/*__attribute__((weak,alias("Default_Handler")))-> weak enable the override of the function ,alias to make alias for the function */
void NMI_Handler()          __attribute__((weak,alias("Default_Handler")));
void H_Fault_Handler()      __attribute__((weak,alias("Default_Handler")));
void MM_Fault_Handler()     __attribute__((weak,alias("Default_Handler")));
void Bus_Fault()            __attribute__((weak,alias("Default_Handler")));
void Usage_Fault_Handler()  __attribute__((weak,alias("Default_Handler")));

/*__attribute__((section(".vectors"))) -> to make a new section called .vectors and put it throught ld at the begining of the flash so the sp at it's place and the IVT
extern uint32_t _stack_top;
uint32_t vectors[] __attribute__((section(".vectors")))=
{
    (uint32_t)&_stack_top,
    (uint32_t)&Reset_Handler,
    (uint32_t)&NMI_Handler,
    (uint32_t)&H_Fault_Handler,
    (uint32_t)&MM_Fault_Handler,
    (uint32_t)&Bus_Fault,
    (uint32_t)&Usage_Fault_Handler,
};

/*symbols not variables so you need to use & */
extern uint32_t _E_text;
extern uint32_t _S_data;
extern uint32_t _E_data;
extern uint32_t _S_bss;
extern uint32_t _E_bss;
void Reset_Handler()
{
    /*copy .data from FLASH to RAM*/
    uint32_t i =0;
    unsigned int Data_size= (uint8_t *)&_E_data -(uint8_t *)&_S_data;
    uint8_t * psrc=(uint8_t *)&_E_text;
    uint8_t * pdes=(uint8_t *)&_S_data;
    for(i =0;i<Data_size;i++)
    {
        *((uint8_t *)pdes)= *((uint8_t *)psrc);
        pdes++;
        psrc++;
    }

    /*init .bss in the RAM*/
    unsigned int Bss_size= (uint8_t *)&_E_bss -(uint8_t *)&_S_bss;
    pdes=(uint8_t *)&_S_bss;
    for(i =0;i<Bss_size;i++)
    {
        *((uint8_t *)pdes++)= (uint8_t)0;
    }
```

LINKER-SCRIPT

```
/*linkerscript cortex-m3
Ahmed Essam
*/


MEMORY
{
    flash(rx) : ORIGIN = 0x8000000, LENGTH = 128k
    sram(rwx) : ORIGIN = 0x20000000, LENGTH = 20k
}

SECTIONS
{
    .text :{
        *(.vectors*)
        *(.text)
        *(.rodata)
        _E_text = . ;
    }>flash

    .data :{
    _S_data = . ;
        *(.data)
        . = ALIGN(4) ;
    _E_data = . ;
    }>sram AT> flash

    .bss :{
    _S_bss = . ;
        *(.bss)
        . = ALIGN(4) ;
    _E_bss = . ;
    . = . + 0x1000 ;
    _stack_top = . ;
    }>sram
}
```

STATE FILE

```
#ifndef _STATE_H
#define _STATE_H

#define STATE_define(_stateFunc_)  void ST_##_stateFunc_()
#define STATE(_stateFunc_) ST_##_stateFunc_

void set_pressure(int val);
//1->high    0->low
 void set_alarm_state(int val);

#endif
```

PRESSURE-SENSOR

```h
#ifndef PRESSURE_SENSOR_H
#define PRESSURE_SENSOR_H

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

typedef enum
{
    PS_READING,
    PS_WAITING
}PS_State;


extern void(*pf_ps_state)(void);
void P_Sensor_init(void);
STATE_define(PS_READING);
STATE_define(PS_WAITING);




#endif
```

```c
#include "PressureSensor.h"


static int pVal=0;
static PS_State ps_status_id;
 void(*pf_ps_state)(void)=P_Sensor_init;

void P_Sensor_init(void)
{
    pVal=0;
    pf_ps_state=STATE(PS_READING);

}

STATE_define(PS_READING)
{
    ps_status_id=PS_READING;
    pVal=getPressureVal();
    pf_ps_state=STATE(PS_WAITING);
    pf_ps_state();
    set_pressure(pVal);

}

STATE_define(PS_WAITING)
{
    ps_status_id=PS_WAITING;
    Delay(1000);
    pf_ps_state=STATE(PS_READING);

}


```

ALGORITHM

```c
#ifndef ALGORITHM_H
#define ALGORITHM_H

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

typedef enum
{
    ALGO_PREESURE_DETECTION,
}ALGO_State;

extern void(*pf_algo_state)(void);
STATE_define(ALGO_PREESURE_DETECTION);
STATE_define(ALGO_HIGH_PRESSURE_DETECTED);


#endif
```

```c
#include "Algorithm.h"


static int pVal=0;
static int pThreshould=20;
static ALGO_State algo_status_id;
 void(*pf_algo_state)(void)=STATE(ALGO_PREESURE_DETECTION);


 STATE_define(ALGO_PREESURE_DETECTION)
 {
    algo_status_id=ALGO_PREESURE_DETECTION;
    pVal>=pThreshould?set_alarm_state(1):set_alarm_state(0);
    pf_algo_state=STATE(ALGO_PREESURE_DETECTION);
 }



 void set_pressure(int val)
 {
    pVal=val;
 }
```

ALARM

```h
#ifndef ALARM_H
#define ALARM_H

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

typedef enum
{
    ALARM_WAITING,
    ALARM_ON,
    ALARM_OFF
}ALARM_State;

extern void(*pf_alarm_state)(void);
void Alarm_init(void);
STATE_define(ALARM_WAITING);
STATE_define(ALARM_ON);
STATE_define(ALARM_OFF);


#endif
```

```c
#include "Alarm.h"


void(*pf_alarm_state)(void);
ALARM_State alarm_state_id;
static int alarm_state=0;
void Alarm_init(void)
{
    pf_alarm_state=STATE(ALARM_WAITING);
    alarm_state=0;
    //hw initializations

}
STATE_define(ALARM_WAITING)
{
    alarm_state_id=ALARM_WAITING;
    alarm_state==0?(pf_alarm_state=STATE(ALARM_OFF)):(pf_alarm_state=STATE(ALARM_ON));
    pf_alarm_state();
}

STATE_define(ALARM_ON)
{
    alarm_state_id=ALARM_ON;
    Set_Alarm_actuator(0);
    Delay(60*1000);
    pf_alarm_state=STATE(ALARM_WAITING);
}
STATE_define(ALARM_OFF)
{
    alarm_state_id=ALARM_OFF;
    Set_Alarm_actuator(1);
    pf_alarm_state=STATE(ALARM_WAITING);
}

void set_alarm_state(int val)
{
    alarm_state=val;
}
```

MAIN

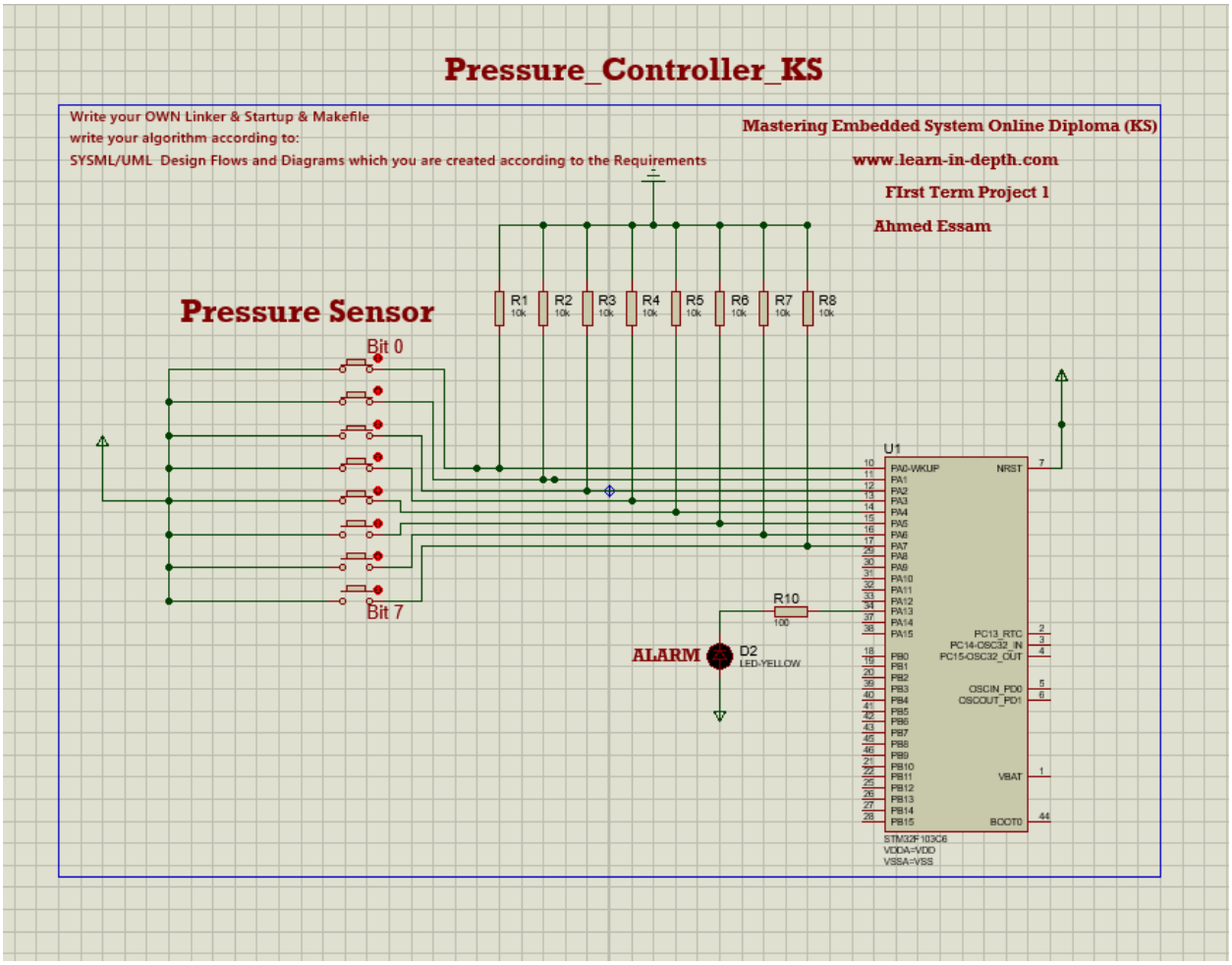
```c
#include <stdint.h>
#include <stdio.h>
#include "driver.h"
#include "PressureSensor.h"
#include "Algorithm.h"
#include "Alarm.h"

int main (){
    GPIO_INITIALIZATION();
    P_Sensor_init();
    Alarm_init();
    while (1)
    {
        //Implement your Design
        pf_ps_state();
        pf_algo_state();
        pf_alarm_state();
    }

}

```

SIMULATION


Pressure_Controller_KS
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
Ahmed Essam
Pressure Sensor
Bit 0
Bit 7
R1 10k
R2 10k
R3 10k
R4 10k
R5 10k
R6 10k
R7 10k
R8 10k
R10 100
ALARM
D2 LED-YELLOW
U1
PA0-WKUP
NRST
PC13_RTC
PC14-OSC32_IN
PC15-OSC32_OUT
OSCIN_PD0
OSCOUT_PD1
VBAT
BOOT0
STM32F103C6
VDDA=VDD
VSSA=VSS