# PRESSURE CONTROLLER

Supervisor: Eng. Keroles
Hassan Abd Eltawab

OCTOBER 2021
LEARN-IN-DEPTH(ONLINE DIPLOMA)
First Term Project

# Table of Contents

# Ch1. Customer Point of View

## 1.1. Case Study

- A" client" expects a software of a system with the following Specification:
  - ▶ 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.
  - ▶ (Optional Feature) keeps track of the measured values
- Assumptions:
  - ▶ The controller set up and shutdown procedures are not modelled
  - ▶ The controller maintenance is not modelled
  - ▶ The pressure sensor never fails
  - ▶ The alarm never fails
  - ▶ The controller never faces power cut

## 1.2. Chosen Design Approch

- Waterfull Method has been chosen for its simplicity.

## 1.3. System Requirements

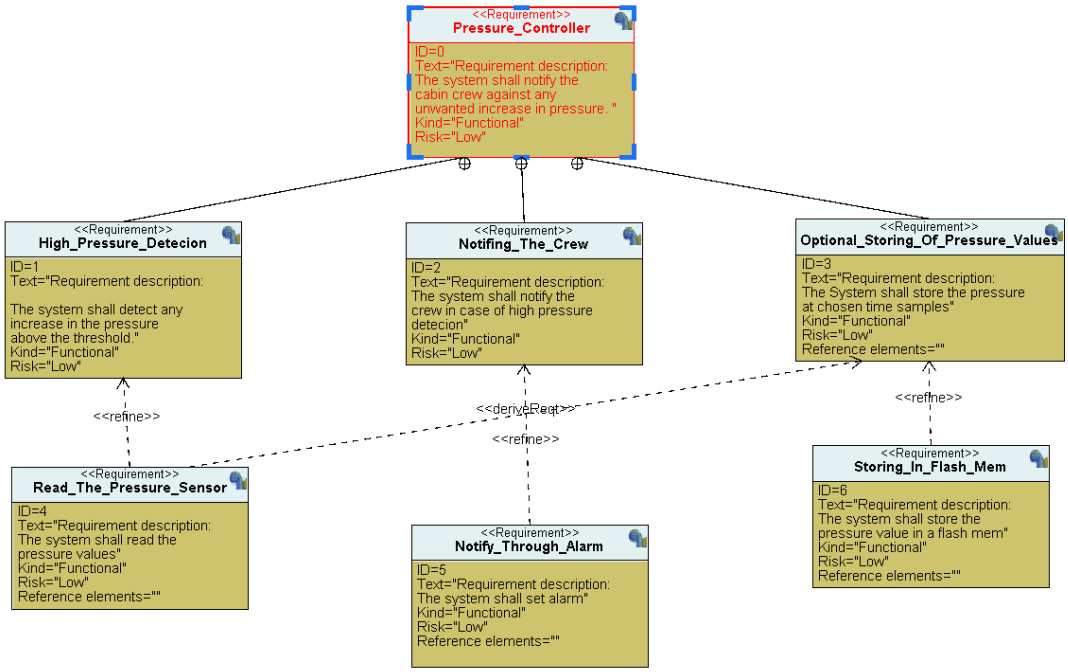



*Figure 1: Requirements Diagram*

# 1.4. System Analysis

## 1.4.1 UML Use Case Diagram

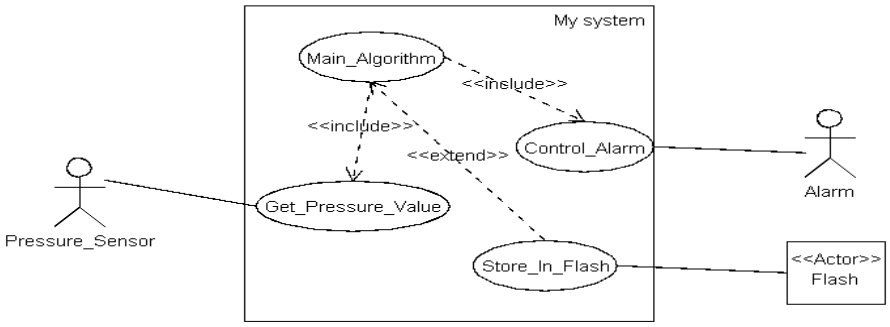



*Figure 2: Use Case Diagram For The System*

## 1.4.2 UML Activity Diagram

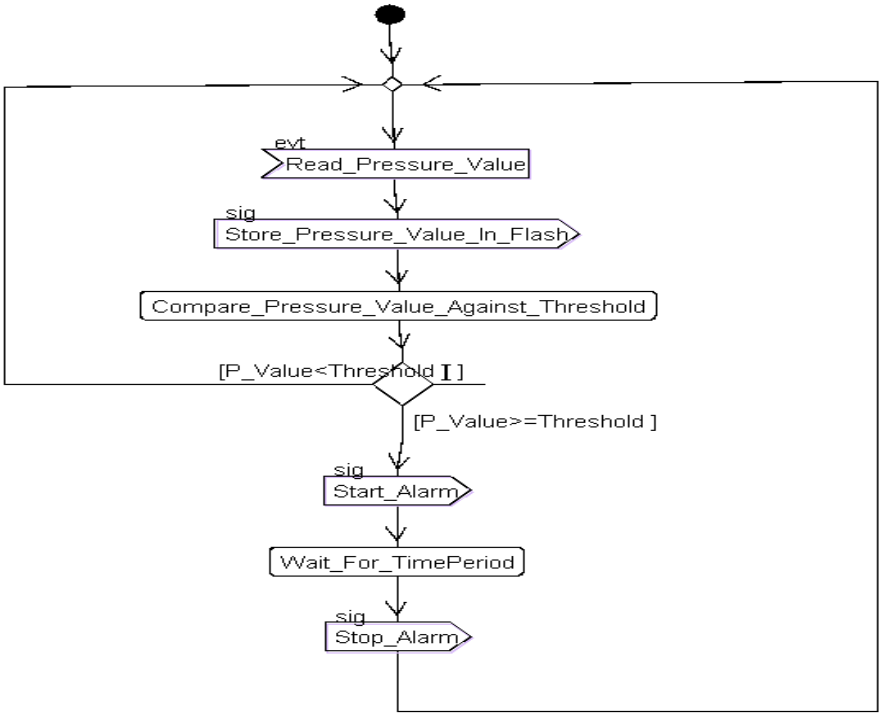



*Figure 3: Activity Diagram For The System*

### 1.4.3. UML Sequence Diagram:

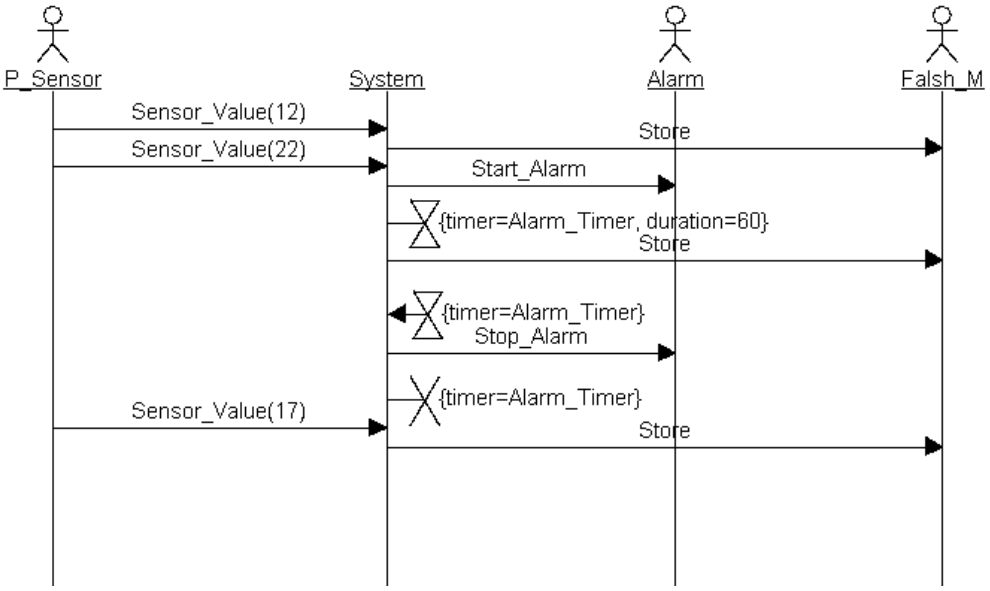



*Figure 4: Built Sequence*

## 1.5. System Design

### 1.5.1. Block Diagram

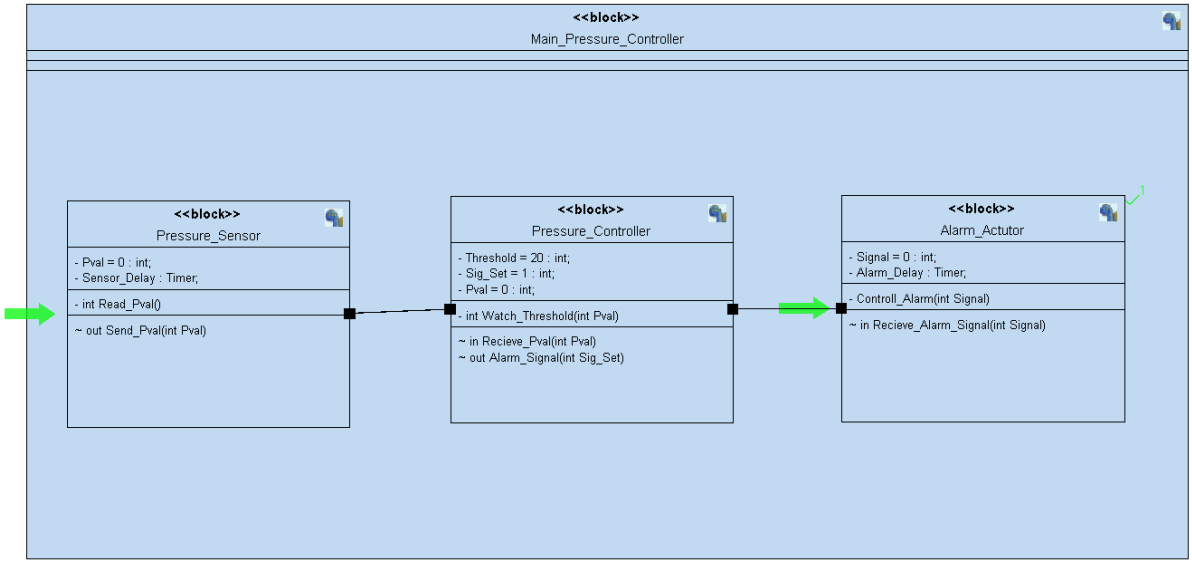



*Figure 5: Block Diagram*

## 1.5.2. State Machines

### 1.5.2.1. State Machines For The Sensor Block

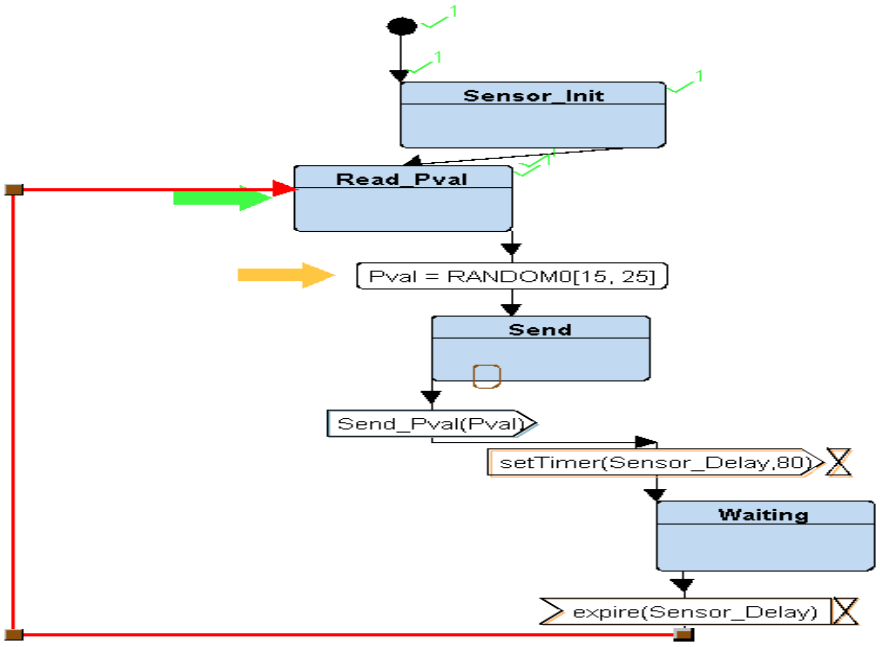



*Figure 6: Sensor State Machines*

### 1.5.2.2. State Machines For The Pressure Controller

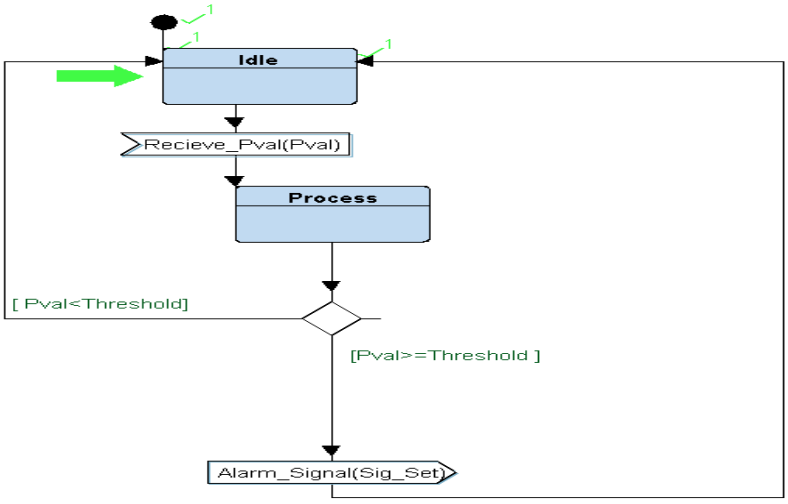



*Figure 7: P Controller State Machines*

### 1.5.2.3. State Machines For Alarm Actuator

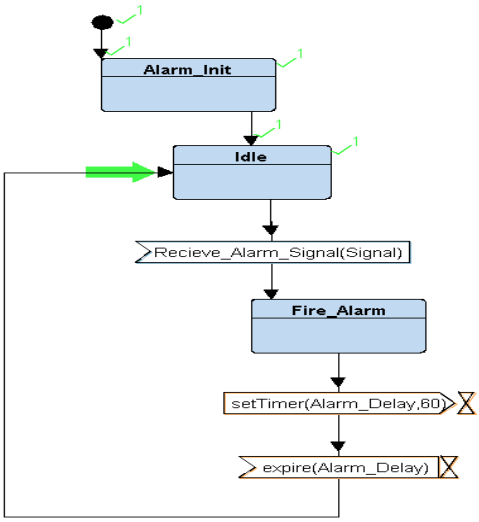
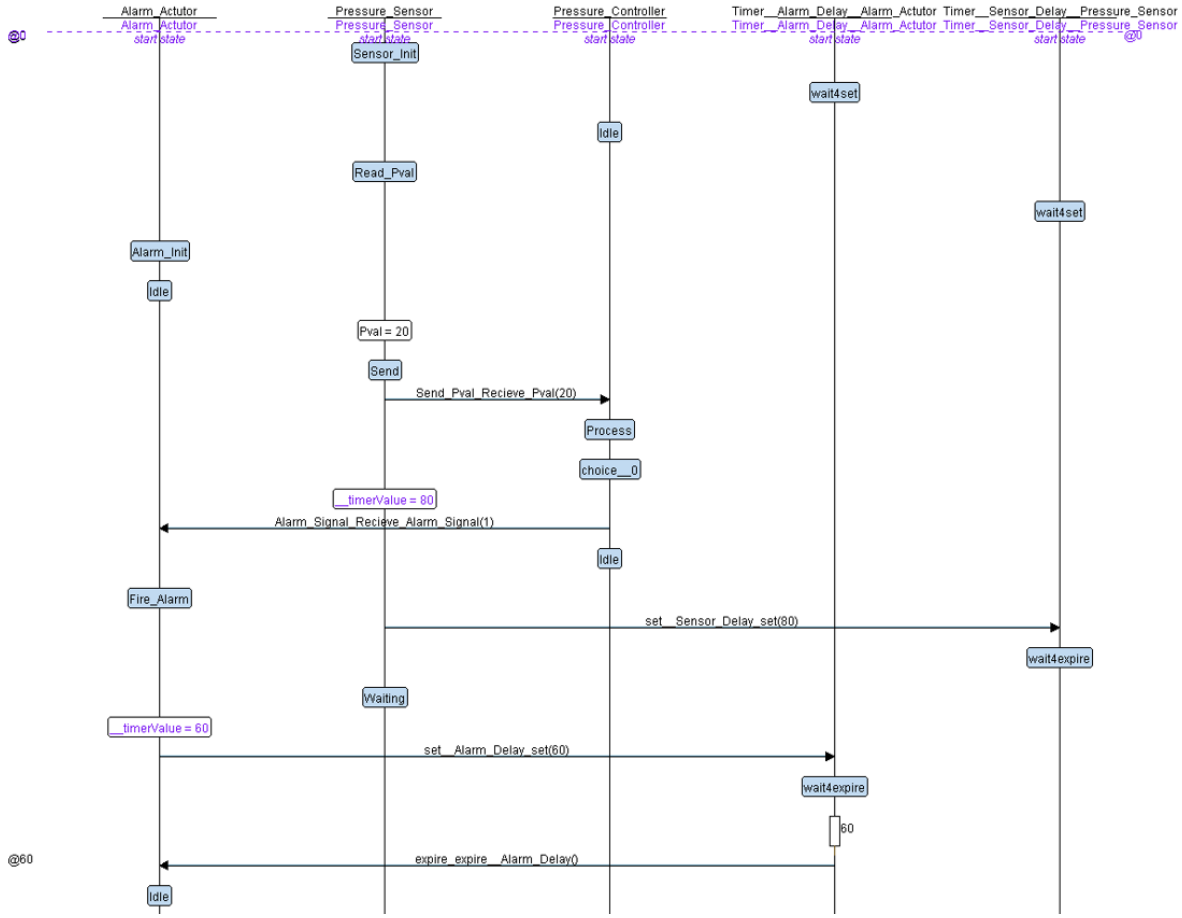
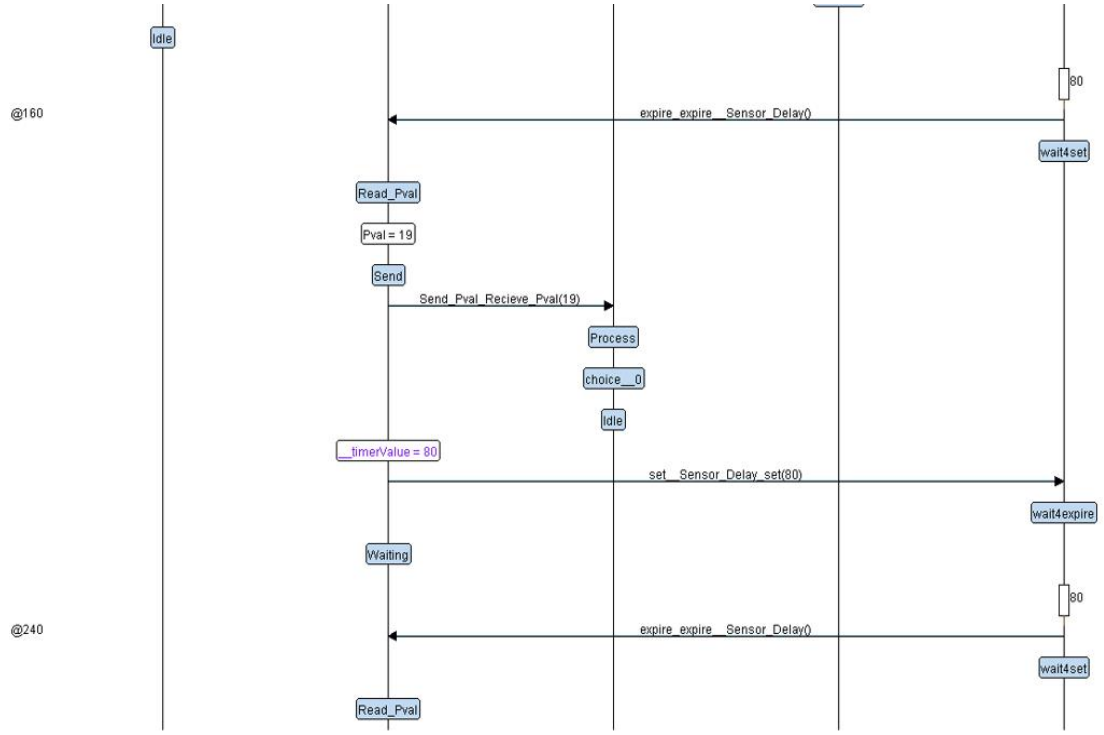



*Figure 8: Alarm State Machines*

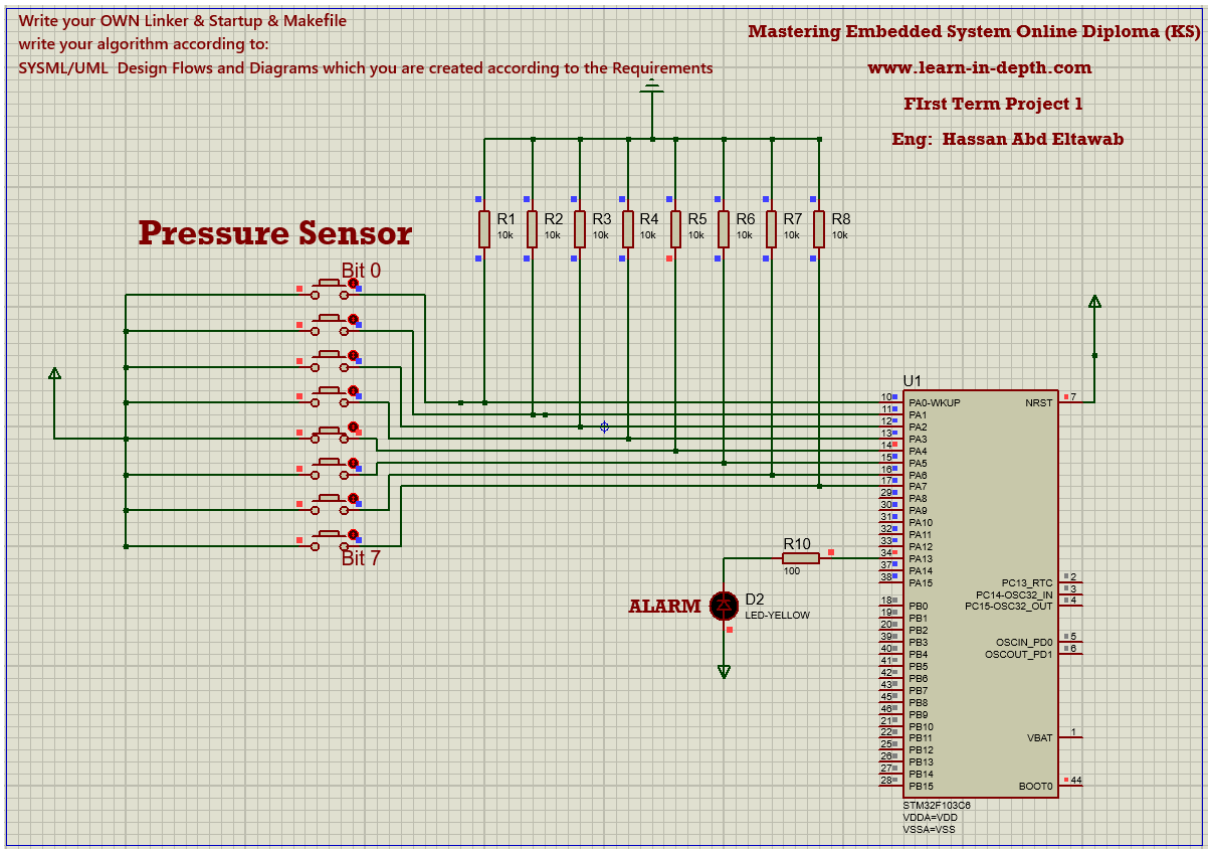
## 1.5.3. Simulated Sequence Diagram



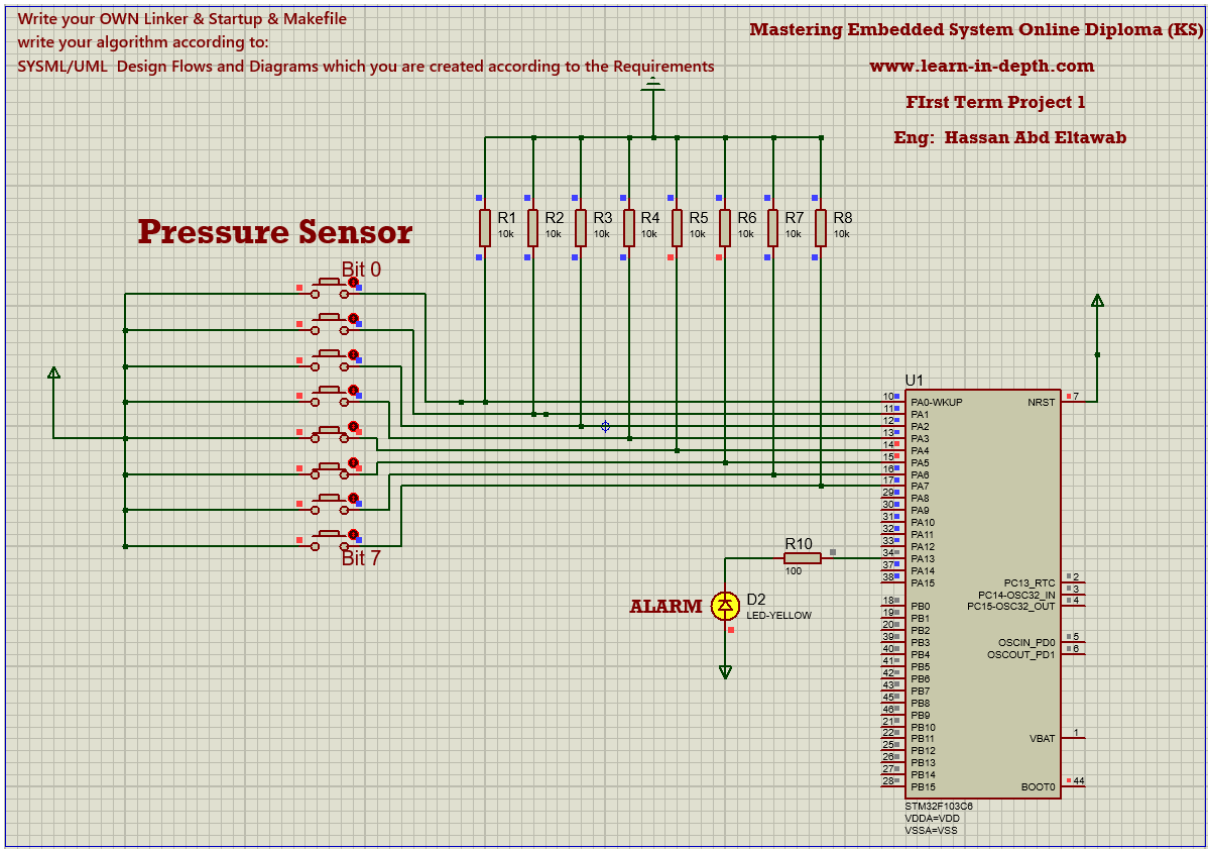



## 1.6 Proteus Simulation

### Case 1. Pressure Below Threshold. Pressure is 20



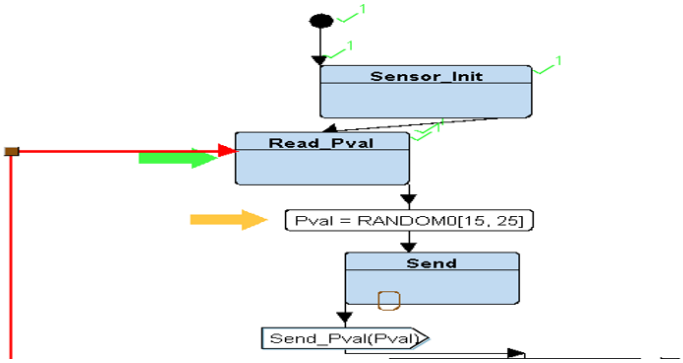
## Case 2. Pressure Above Threshold Pressure is 48




# Ch2. Technical Point of View

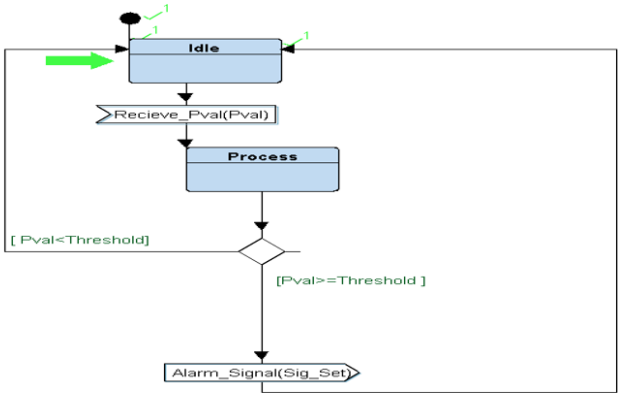
## 2.1. State Machines Accombined with Codes:

### 2.1.1. Pressure Sensor Module:




```
10 ▼ void PSensor_Read_Pval(){
11       Pval=getPressureVal();
12       PSensor_state=PSensor_Send_Pval;
13   }
14   /*Sending State*/
15 ▼ void PSensor_Send_Pval(){
16       PController_Receive_Pval(Pval);
17       PSensor_state=PSensor_Read_Pval;
18   }
```

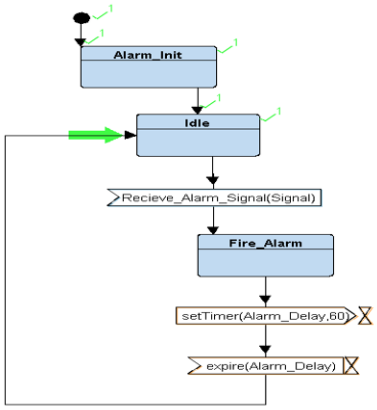
## 2.1.2 Pressure Controller Module




```
/*The Idle State*/
void PController_Idle(){
    PController_state=PController_Idle;
}
/*The Processing  State*/
void PController_Process(){

        if(Pval>=20){
            /*Sending Signal To Alarm Module*/
            Alarm_Signal(1);
            PController_state=PController_Idle;
        }else{
            PController_state=PController_Idle;
        }
}
```

## 2.1.3. Alarm Actuator Module




```
void Alarm_Idle(){
    Alarm_State=Alarm_Idle;
}
/*Fire Alarm State Upon Recieving Signal*/
void Alarm_Fire(){
        Set_Alarm_actuator(0);
        //_delay_ms(60000);
        Delay(1000000);
        Set_Alarm_actuator(1);
        Alarm_State=Alarm_Idle;
}
void Alarm_Signal(int signal){
    if (signal){
        Alarm_State=Alarm_Fire;
    }
}
```

## 2.2. Final Image Mapfile

```
Memory Configuration

Name             Origin             Length             Attributes
flash            0x0000000008000000 0x0000000000020000 xr
sram             0x0000000020000000 0x0000000000005000 xrw
*default*        0x0000000000000000 0xffffffffffffffff

Linker script and memory map


.text           0x0000000008000000      0x31c
 *(.vectors*)
 .vectors       0x0000000008000000       0x1c startup.o
                0x0000000008000000                vectors
 *(.rodata*)
                0x000000000800031c                _E_text = .

.data           0x0000000020000000        0x0 load address 0x000000000800031c
                0x0000000020000000                _S_DATA = .
 *(.data)
 .data          0x0000000020000000        0x0 startup.o
 .data          0x0000000020000000        0x0 P_Controller.o
 .data          0x0000000020000000        0x0 Alarm_Act.o
 .data          0x0000000020000000        0x0 main.o
 .data          0x0000000020000000        0x0 P_Sensor.o
 .data          0x0000000020000000        0x0 driver.o
                0x0000000020000000                . = ALIGN (0x4)
                0x0000000020000000                _E_DATA = .

.bss            0x0000000020000000     0x1014 load address 0x000000000800031c
                0x0000000020000000                _S_bss = .
 *(.bss*)
 .bss           0x0000000020000000        0x0 startup.o
 .bss           0x0000000020000000        0x0 P_Controller.o
 .bss           0x0000000020000000        0x0 Alarm_Act.o
 .bss           0x0000000020000000        0x0 main.o
 .bss           0x0000000020000000        0x0 P_Sensor.o
 .bss           0x0000000020000000        0x0 driver.o
                0x0000000020000000                . = ALIGN (0x4)
                0x0000000020000000                _E_bss = .
```

## 2.3. Final Image Sections

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

Pressure_Controller_Project.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000031c  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  20000000  0800031c  00020000  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001014  20000000  0800031c  00020000  2**2
                  ALLOC
  3 .debug_info   00003381  00000000  00000000  00020000  2**0
```

## 2.4. Final Image Symbols

```
$ arm-none-eabi-nm.exe Pressure_Controller_Project.elf
20000000 B _E_bss
20000000 D _E_DATA
0800031c T _E_text
20000000 B _S_bss
20000000 D _S_DATA
20001000 B _stack_top
0800014c T Alarm_Fire
08000130 T Alarm_Idle
08000124 T Alarm_Init
08000178 T Alarm_Signal
2000100c B Alarm_State
0800001c W Bus_Fault
0800001c T Default_Handler
08000248 T Delay
08000268 T getPressureVal
080002bc T GPIO_INITIALIZATION
0800001c W H_fault_Handler
080001d8 T main
0800001c W MM_fault_Handler
0800001c W NMI_Handler
080000d8 T PController_Idle
080000f4 T PController_Process
080000ac T PController_Receive_Pval
20001004 B PController_state
08000200 T PSensor_Read_Pval
08000224 T PSensor_Send_Pval
20001010 B PSensor_state
20001000 B Pval
08000028 T Reset_Handler
08000280 T Set_Alarm_actuator
080001a0 T Setup
20001008 B signal
0800001c W Usage_Fault_Handler
08000000 T vectors
```