Mastering Embedded System Online Diploma

www.learn-in-depth.com

First Term (Final Project 1)

Eng. Arsany Ashraf Mounir

My Profile:

https://www.learn-in-depth.com/online-diploma/arsanyashrafmounir@gmail.com

Pressure Detection Project

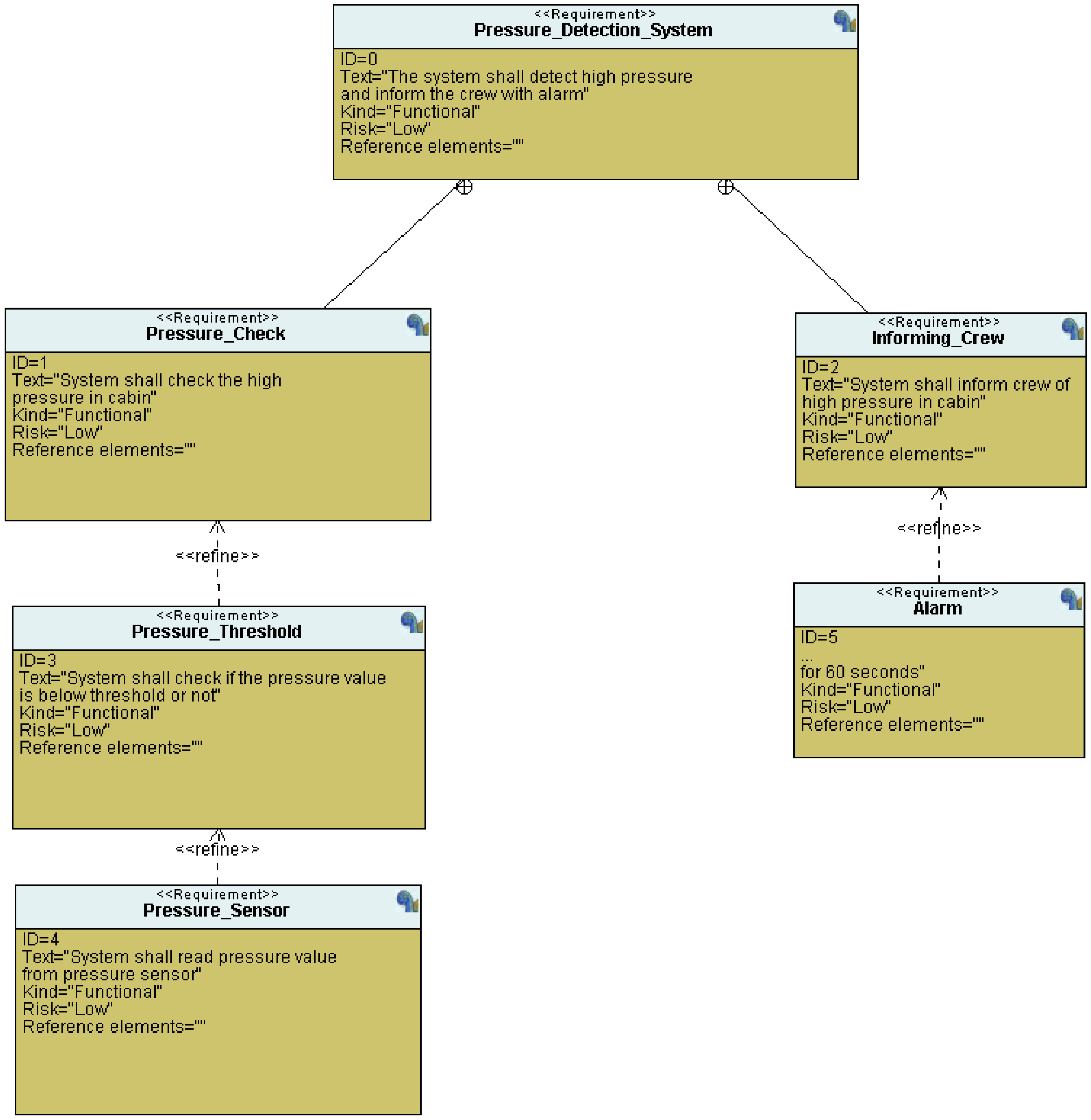
Case Study:-

- A Pressure controller that monitors the pressure level in a cabin with an alarm
- If the pressure exceeds **20 bars** in the cabin the alarm goes off for **60 seconds**

Assumptions:-

- Controller set up and shutdown procedures are not modeled
- Controller maintenance is not modeled
- Pressure sensor used never fails
- Alarm never fails
- No power cut

Requirement diagram:-

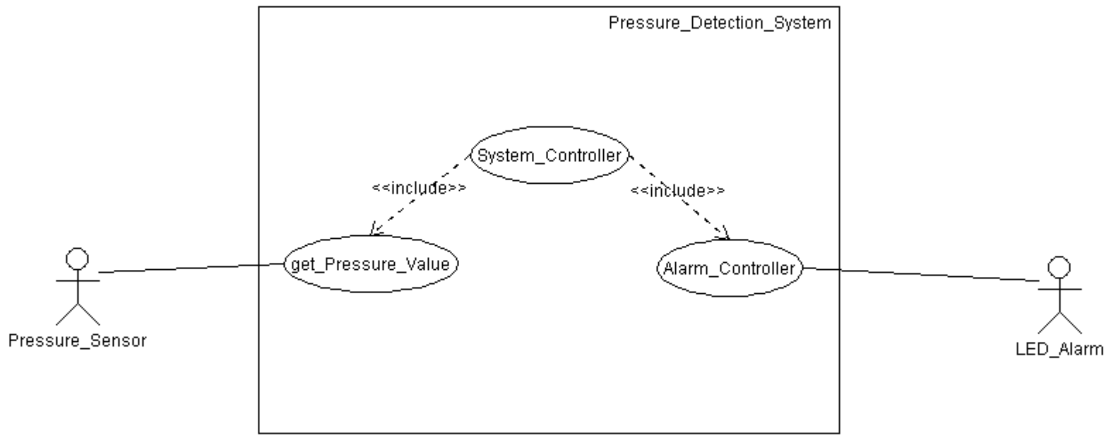


Space Exploration:-

- This is a simple project , it only needs one ECU which will be **STM32**

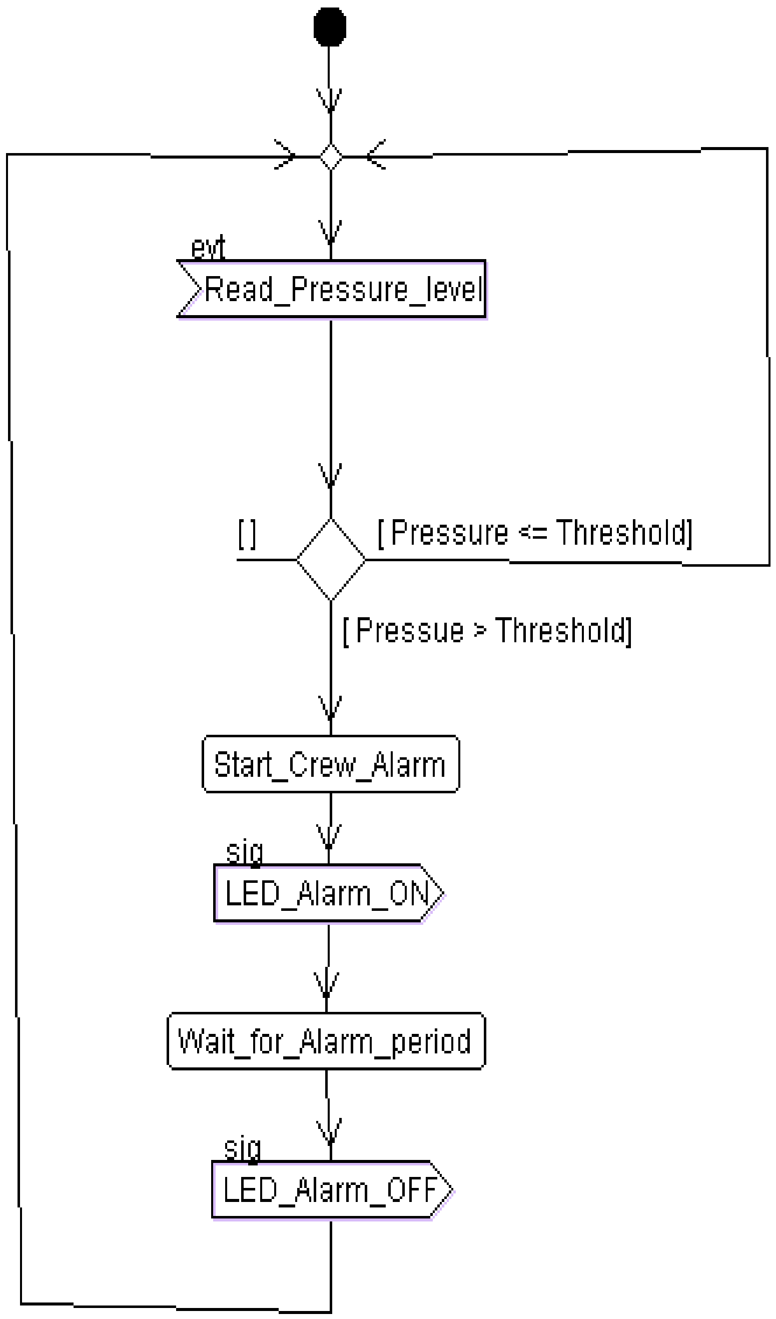
System Analysis:-

- **Use Case Diagram:-**

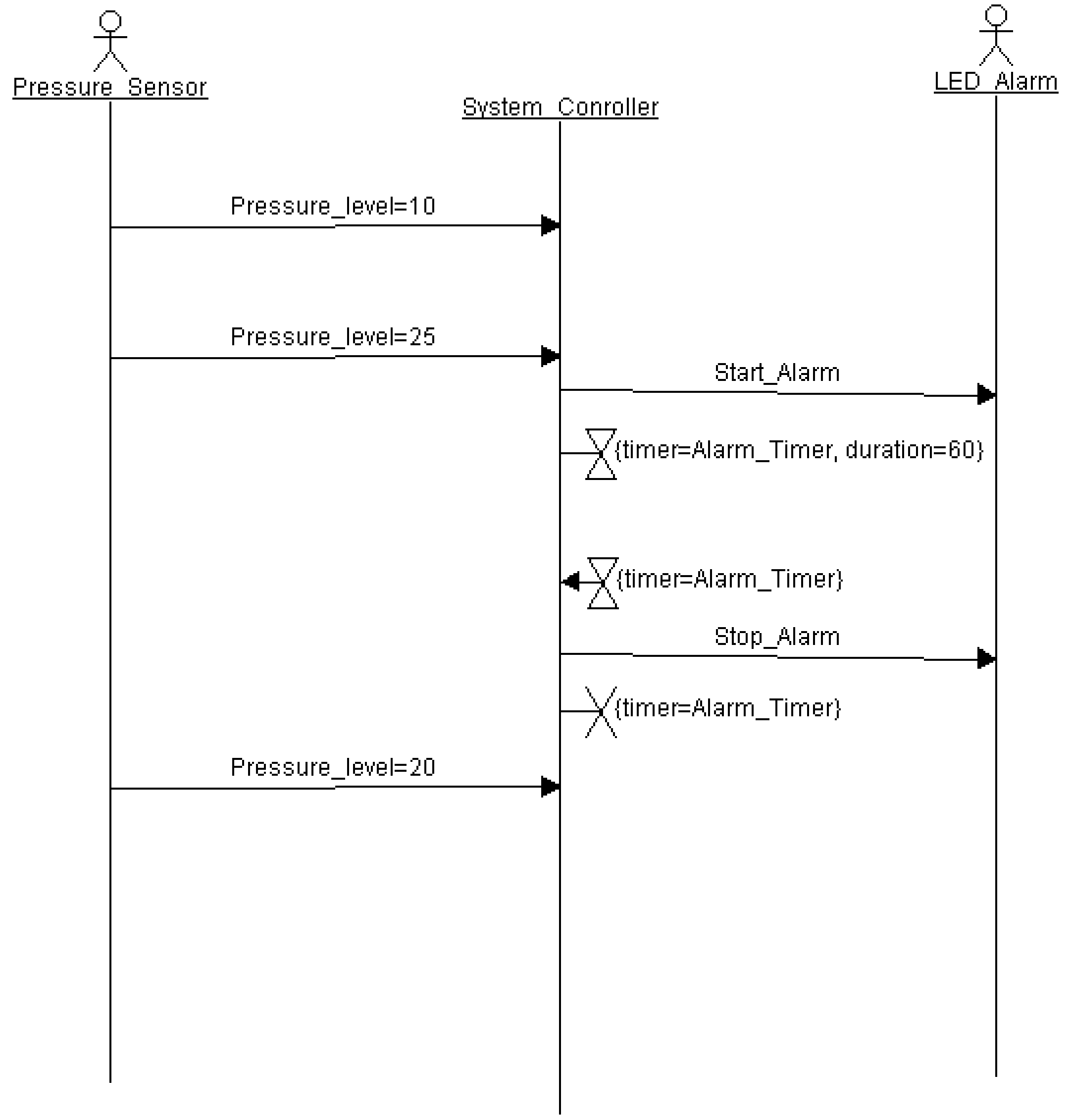


1. get_Pressure_Value reads the pressure level from Pressure Sensor
2. System_Controller compares the pressure level with the threshold "20 bar" , if it exceeds the threshold it send a signal to Alarm_Controller
3. Alarm_Controller manges the LED_Alarm whether to turn it on for 60 second or to turn it off

- **Activity Diagram:-**



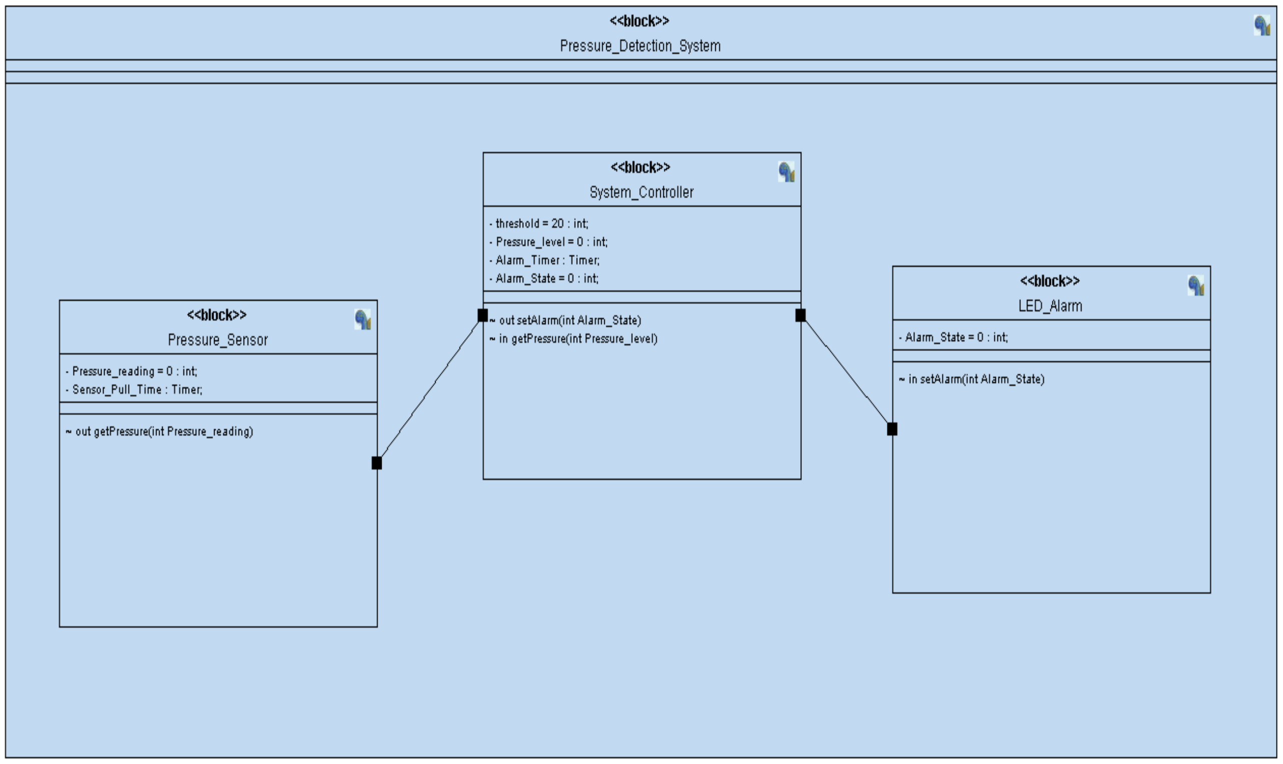
- **Sequence Diagram:-**



Here we notice the when pressure_level=20 nothing happens , since in algorithm it will be defined as less than threshold case

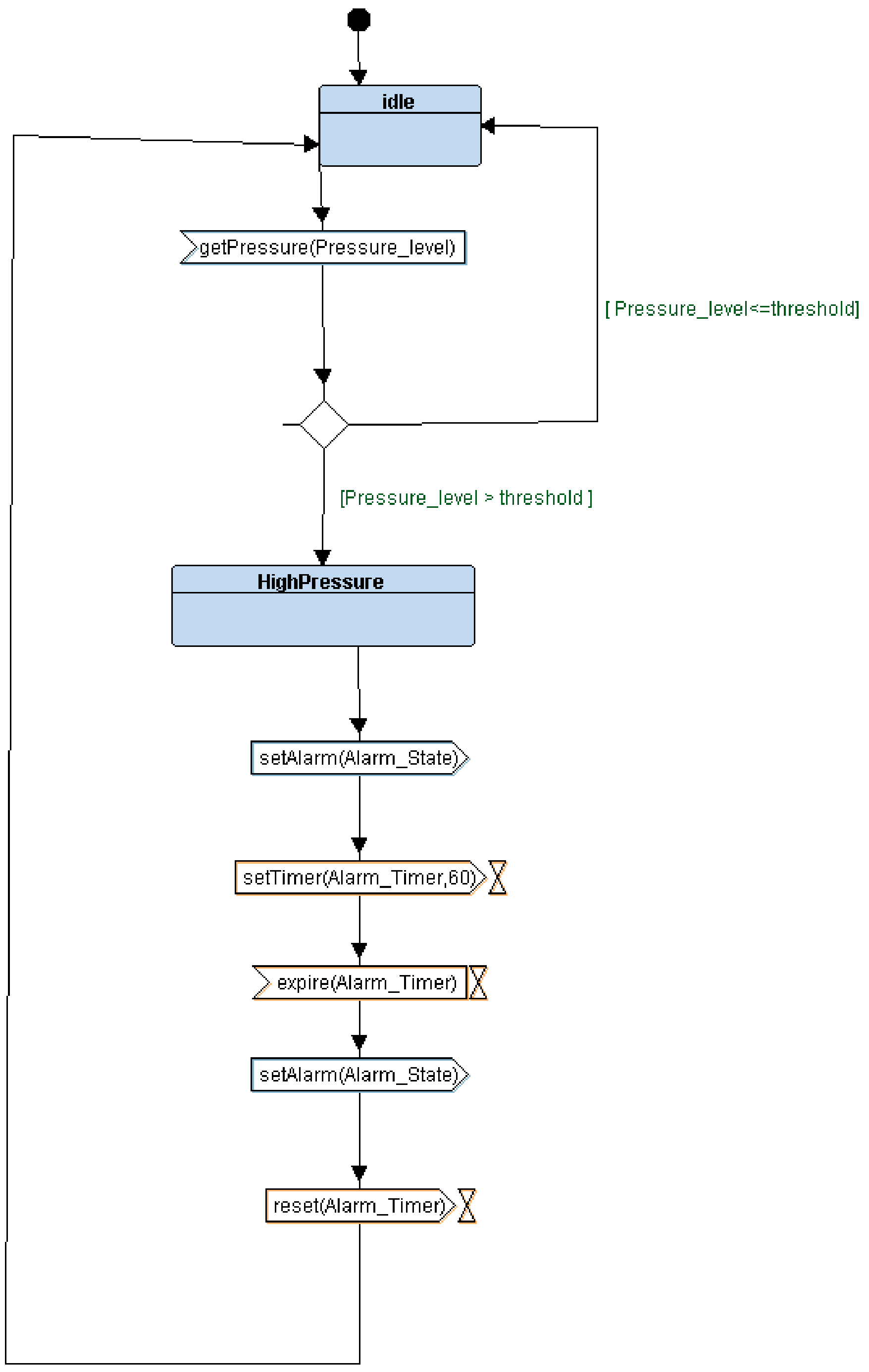
System Design:-

- **Block Diagram:-**

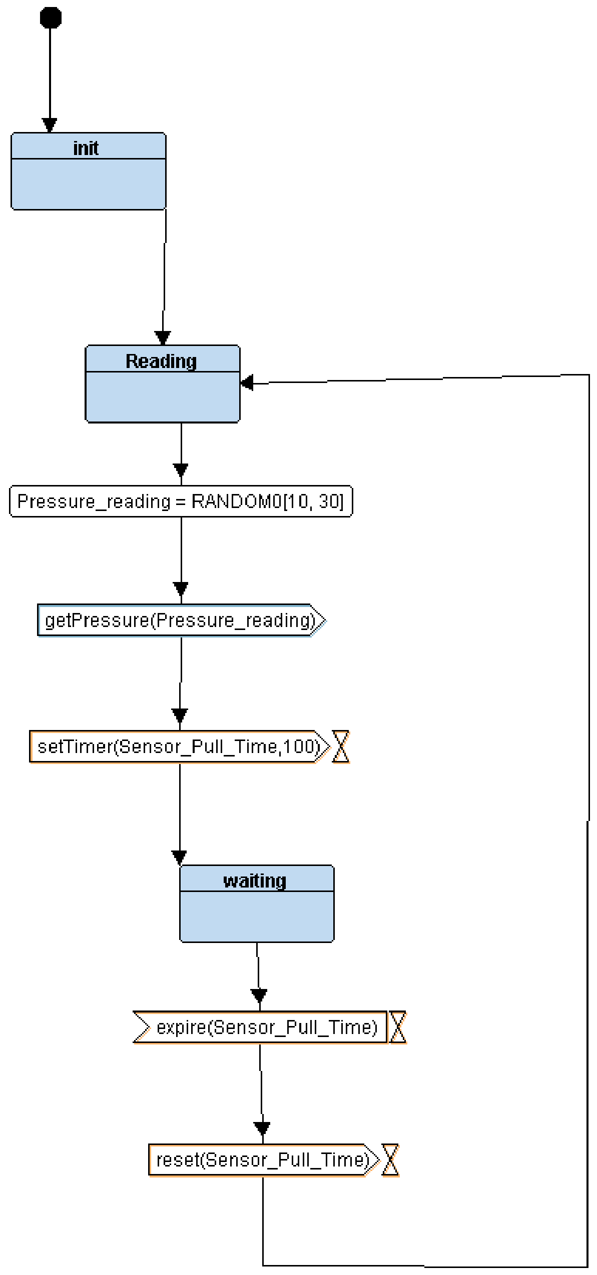


<<block>>
Pressure_Detection_System
<<block>>
System_Controller
- threshold = 20 : int;
- Pressure_level = 0 : int;
- Alarm_Timer : Timer;
- Alarm_State = 0 : int;
~ out setAlarm(int Alarm_State)
~ in getPressure(int Pressure_level)
<<block>>
Pressure_Sensor
- Pressure_reading = 0 : int;
- Sensor_Pull_Time : Timer;
~ out getPressure(int Pressure_reading)
<<block>>
LED_Alarm
- Alarm_State = 0 : int;
~ in setAlarm(int Alarm_State)

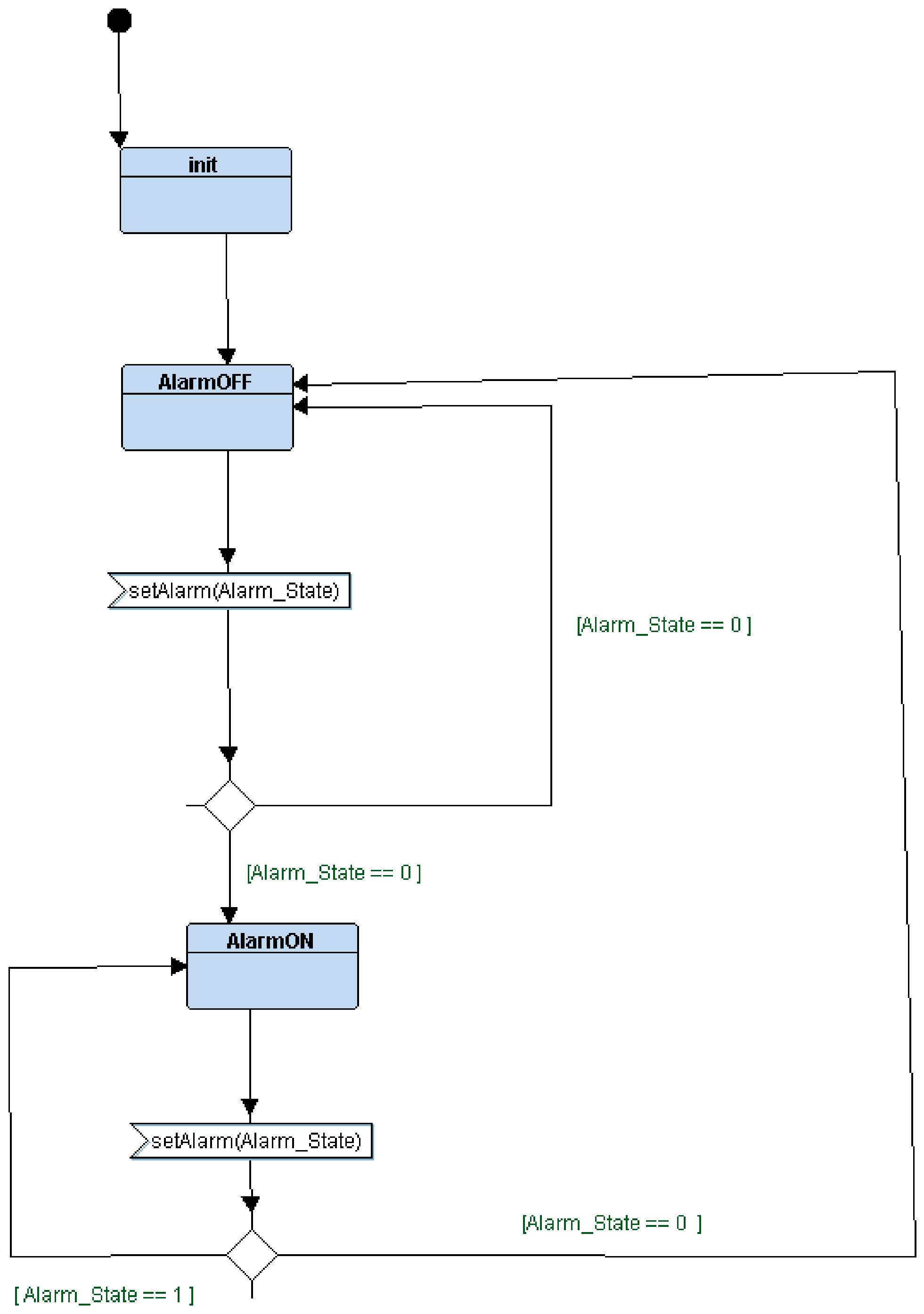
- **<u>State machine System_Controller:-</u>**



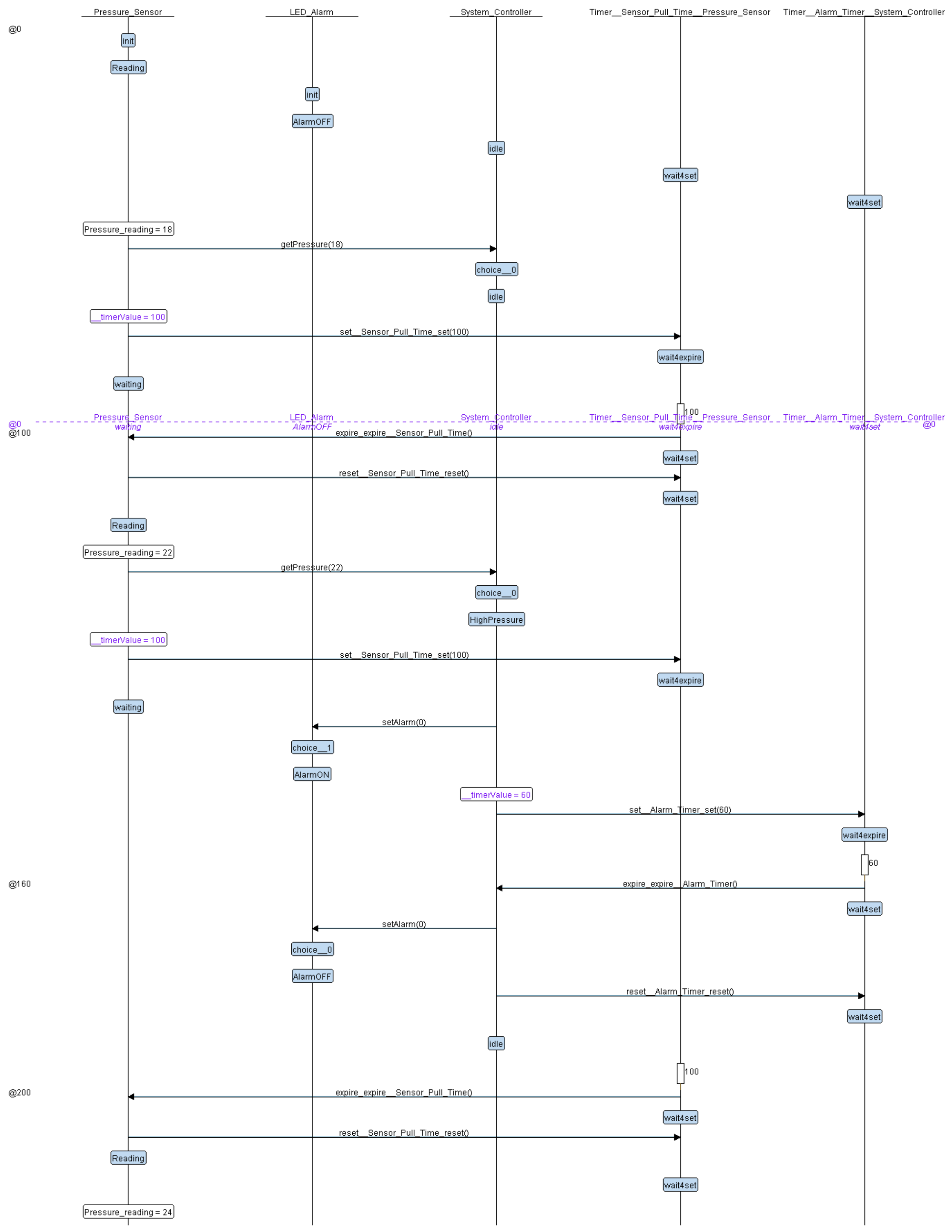
- **State machine Pressure_Sensor:-**


init
Reading
Pressure_reading = RANDOM0[10, 30]
getPressure(Pressure_reading)
setTimer(Sensor_Pull_Time,100)
waiting
expire(Sensor_Pull_Time)
reset(Sensor_Pull_Time)

- **<u>State machine LED_Alarm:-</u>**

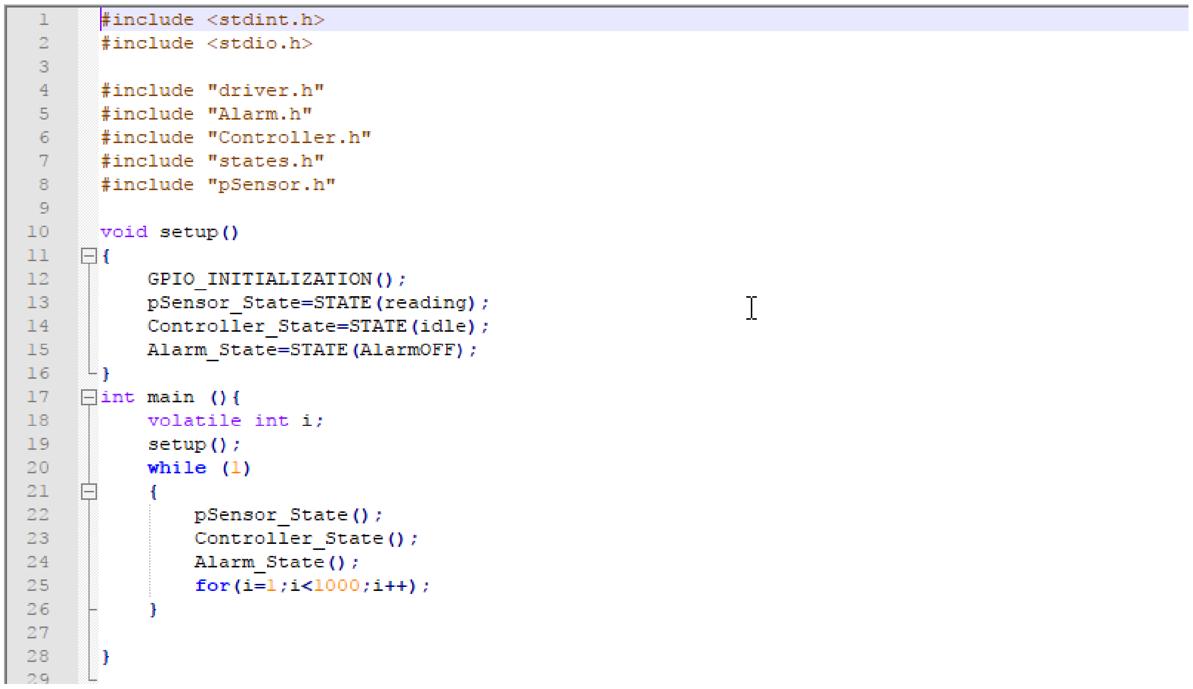


- **<u>Verification of logic using trace:-</u>**



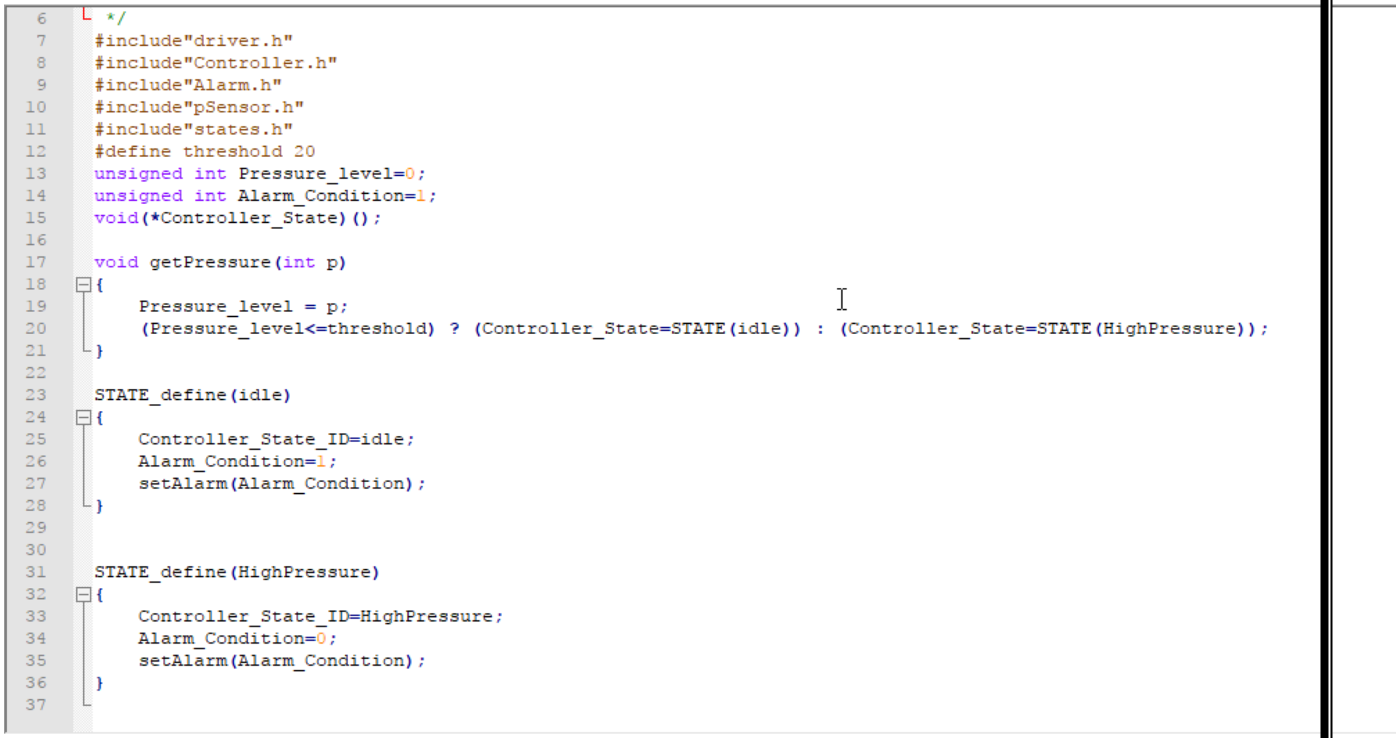
Software of each module:-

- **Main.c**



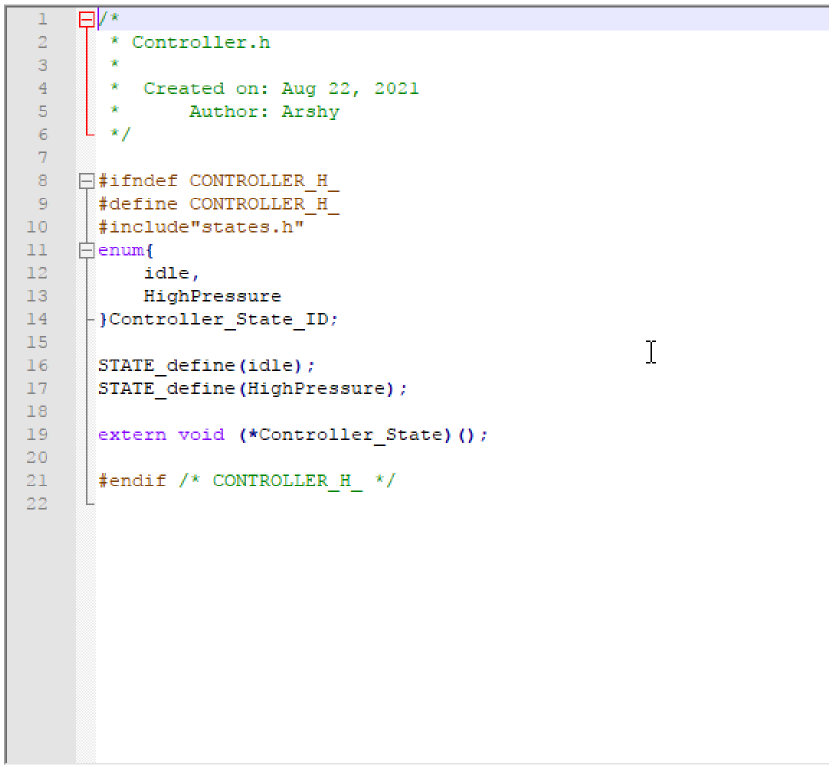
```c
#include <stdint.h>
#include <stdio.h>

#include "driver.h"
#include "Alarm.h"
#include "Controller.h"
#include "states.h"
#include "pSensor.h"

void setup()
{
    GPIO_INITIALIZATION();
    pSensor_State=STATE(reading);
    Controller_State=STATE(idle);
    Alarm_State=STATE(AlarmOFF);
}
int main (){
    volatile int i;
    setup();
    while (1)
    {
        pSensor_State();
        Controller_State();
        Alarm_State();
        for(i=1;i<1000;i++);
    }

}
```

- **Controller.c**



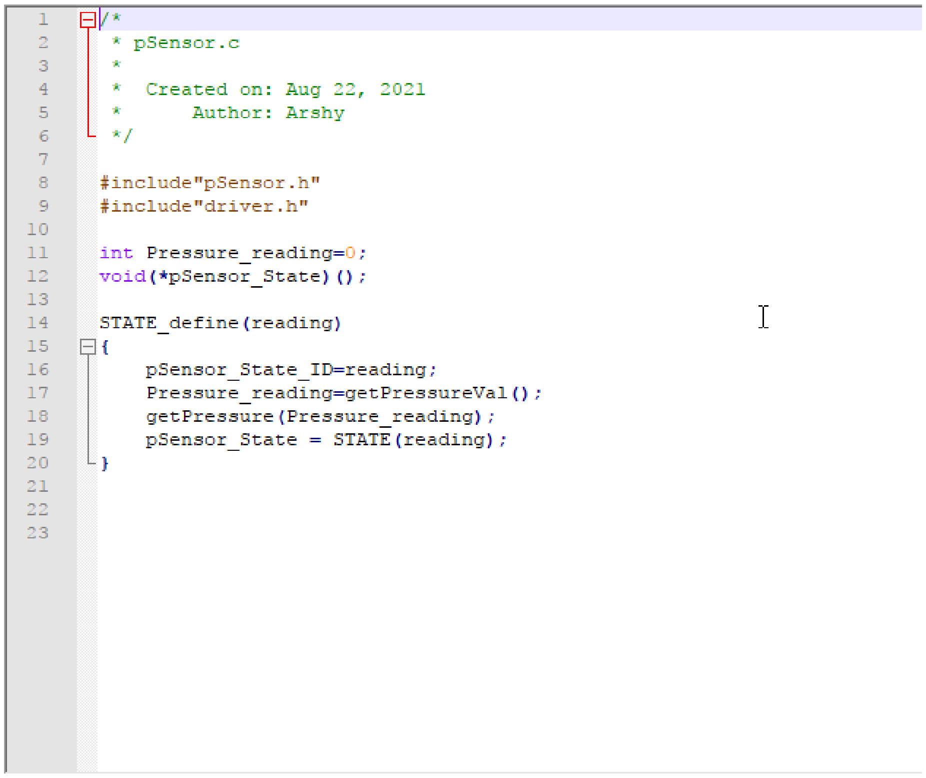
```c
 */
#include"driver.h"
#include"Controller.h"
#include"Alarm.h"
#include"pSensor.h"
#include"states.h"
#define threshold 20
unsigned int Pressure_level=0;
unsigned int Alarm_Condition=1;
void(*Controller_State)();

void getPressure(int p)
{
    Pressure_level = p;
    (Pressure_level<=threshold) ? (Controller_State=STATE(idle)) : (Controller_State=STATE(HighPressure));
}

STATE_define(idle)
{
    Controller_State_ID=idle;
    Alarm_Condition=1;
    setAlarm(Alarm_Condition);
}


STATE_define(HighPressure)
{
    Controller_State_ID=HighPressure;
    Alarm_Condition=0;
    setAlarm(Alarm_Condition);
}
```

- **Controller.h**



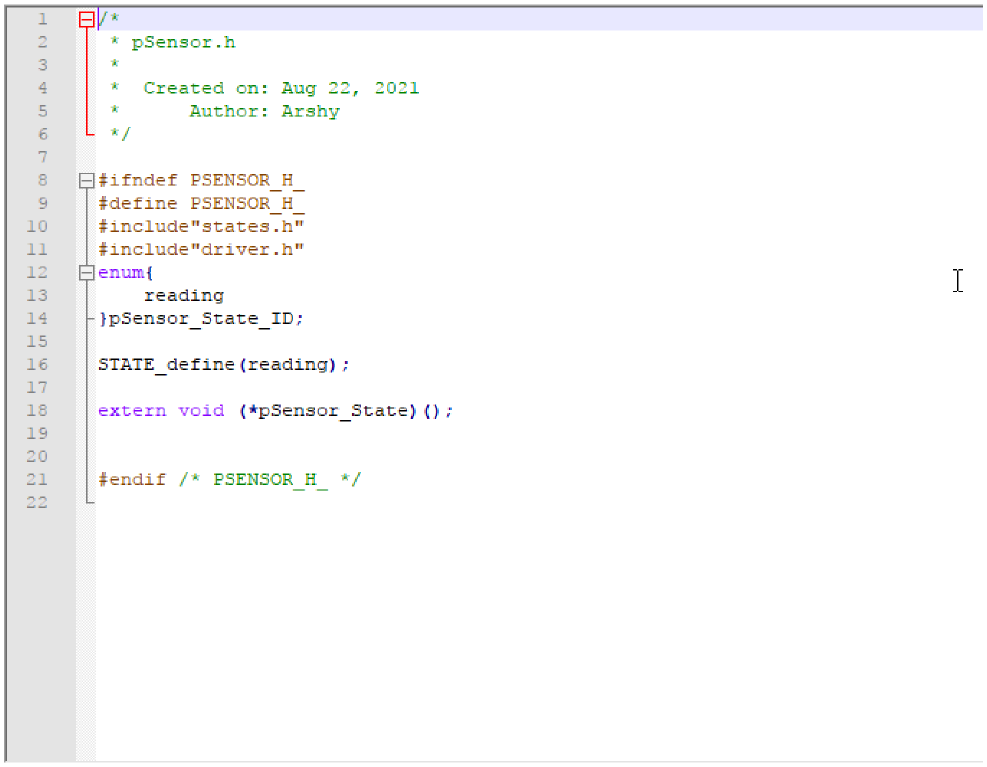
```
/*
 * Controller.h
 *
 *  Created on: Aug 22, 2021
 *      Author: Arshy
 */

#ifndef CONTROLLER_H_
#define CONTROLLER_H_
#include"states.h"
enum{
    idle,
    HighPressure
}Controller_State_ID;

STATE_define(idle);
STATE_define(HighPressure);

extern void (*Controller_State)();

#endif /* CONTROLLER_H_ */

```

- **Pressure Sensor.c**



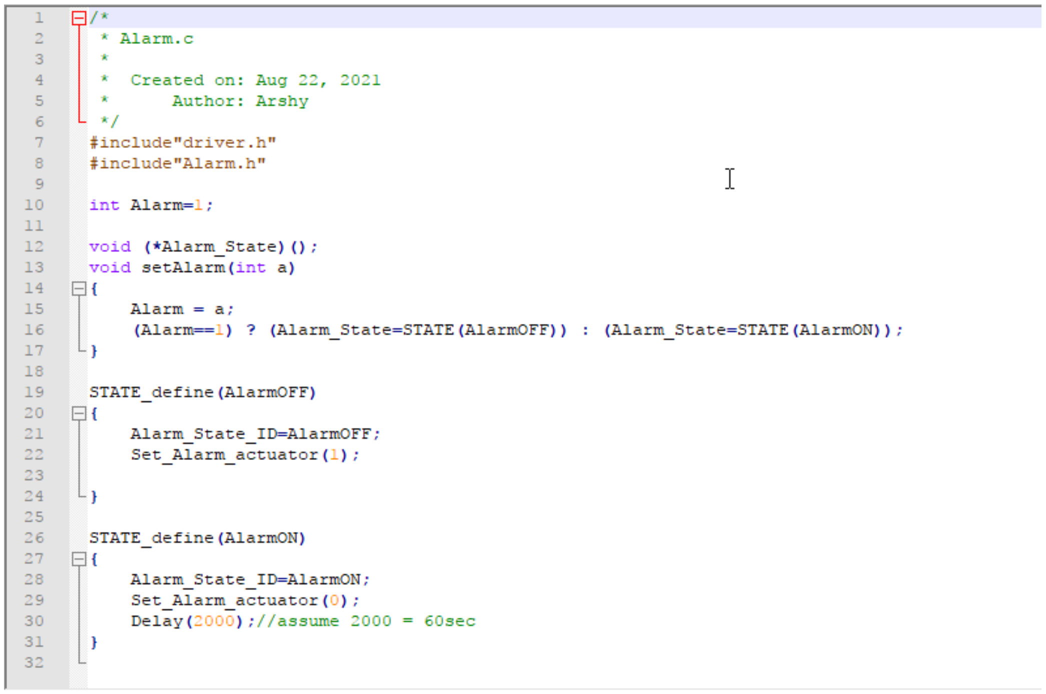
```
/*
 * pSensor.c
 *
 *  Created on: Aug 22, 2021
 *      Author: Arshy
 */

#include"pSensor.h"
#include"driver.h"

int Pressure_reading=0;
void(*pSensor_State)();

STATE_define(reading)
{
    pSensor_State_ID=reading;
    Pressure_reading=getPressureVal();
    getPressure(Pressure_reading);
    pSensor_State = STATE(reading);
}



```

- **Pressure Sensor.h**



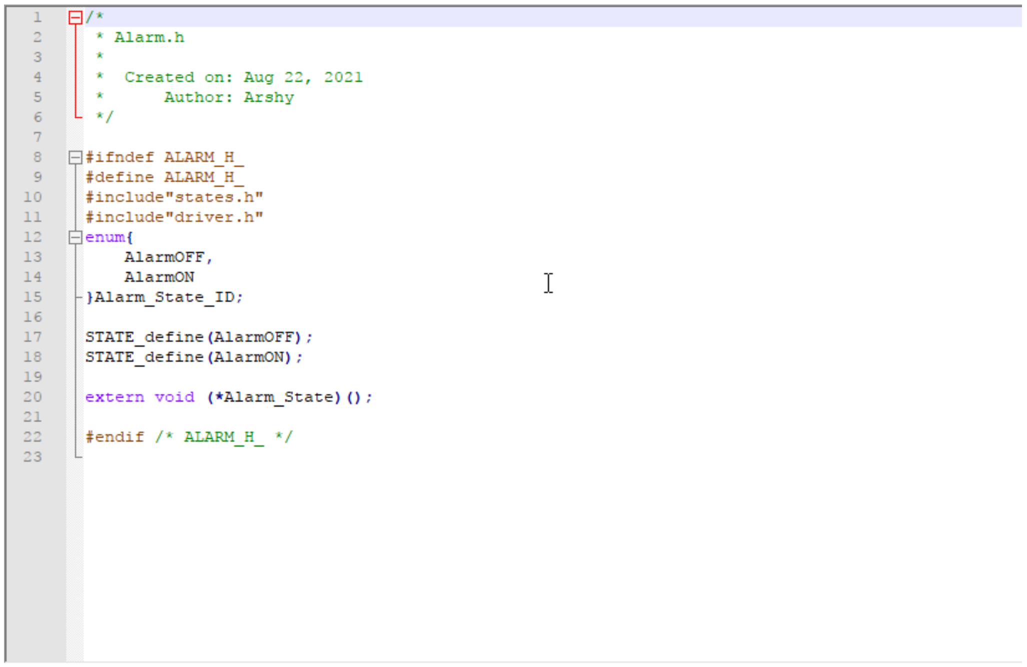
```c
/*
 * pSensor.h
 *
 *  Created on: Aug 22, 2021
 *      Author: Arshy
 */

#ifndef PSENSOR_H_
#define PSENSOR_H_
#include"states.h"
#include"driver.h"
enum{
    reading
}pSensor_State_ID;

STATE_define(reading);

extern void (*pSensor_State)();


#endif /* PSENSOR_H_ */

```

- **Alarm.c**



```c
/*
 * Alarm.c
 *
 *  Created on: Aug 22, 2021
 *      Author: Arshy
 */
#include"driver.h"
#include"Alarm.h"

int Alarm=1;

void (*Alarm_State)();
void setAlarm(int a)
{
    Alarm = a;
    (Alarm==1) ? (Alarm_State=STATE(AlarmOFF)) : (Alarm_State=STATE(AlarmON));
}

STATE_define(AlarmOFF)
{
    Alarm_State_ID=AlarmOFF;
    Set_Alarm_actuator(1);

}

STATE_define(AlarmON)
{
    Alarm_State_ID=AlarmON;
    Set_Alarm_actuator(0);
    Delay(2000);//assume 2000 = 60sec
}

```

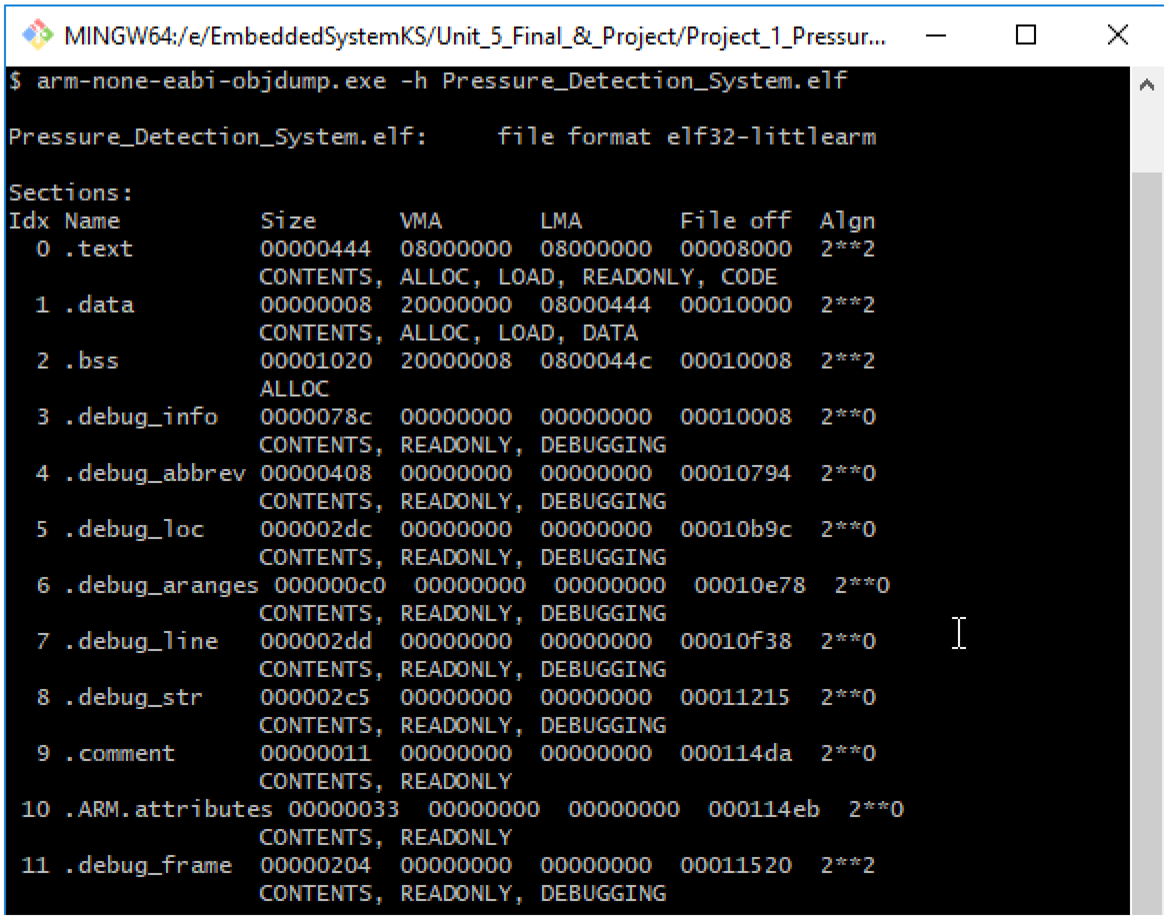
- **Alarm.h**



```
/*
 * Alarm.h
 *
 *  Created on: Aug 22, 2021
 *      Author: Arshy
 */

#ifndef ALARM_H_
#define ALARM_H_
#include"states.h"
#include"driver.h"
enum{
    AlarmOFF,
    AlarmON
}Alarm_State_ID;

STATE_define(AlarmOFF);
STATE_define(AlarmON);

extern void (*Alarm_State)();

#endif /* ALARM_H_ */

```

Software Analysis:-

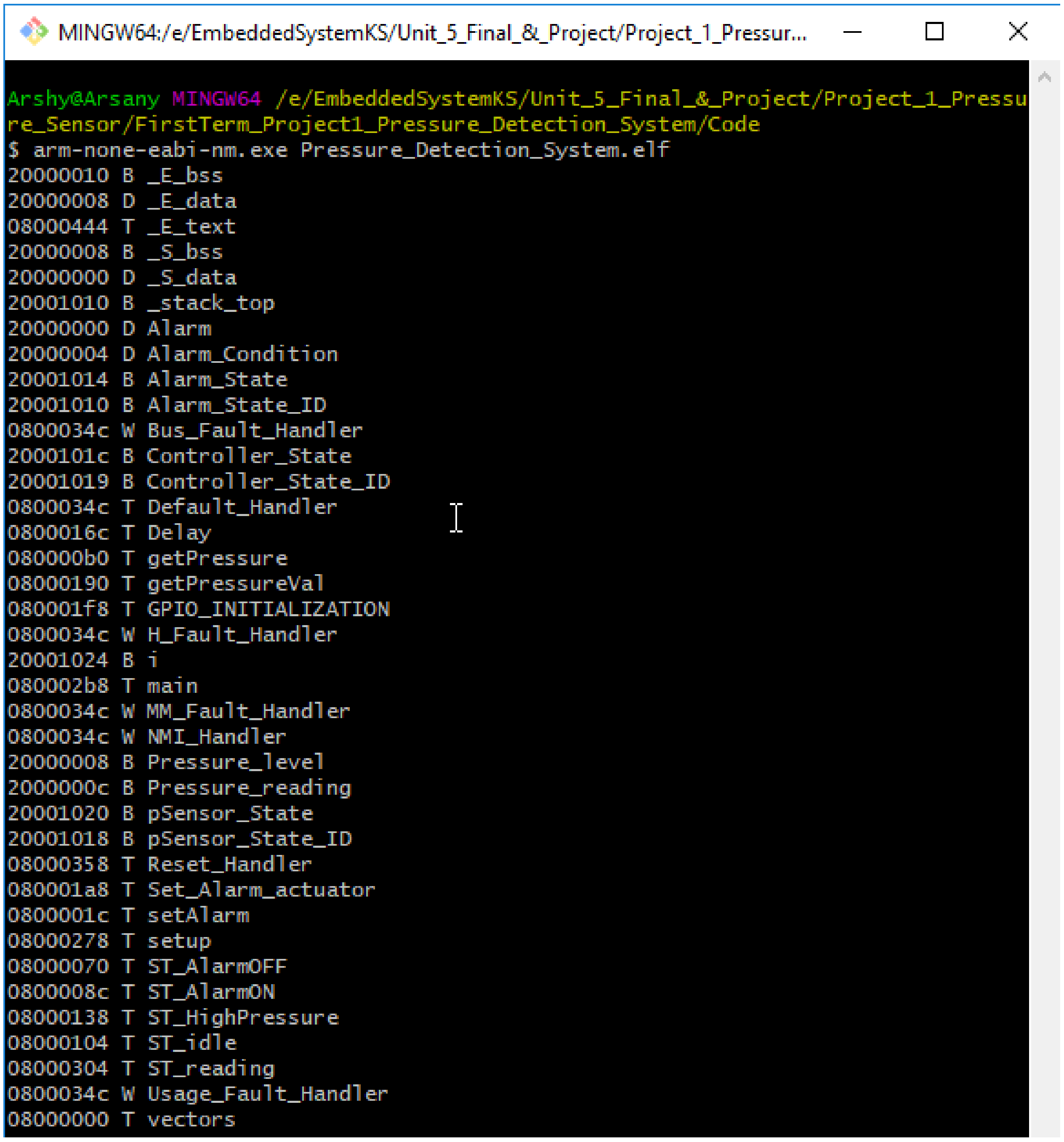
- **Section Table**



MINGW64:/e/EmbeddedSystemKS/Unit_5_Final_&_Project/Project_1_Pressur...

```
$ 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         00000444  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  20000000  08000444  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001020  20000008  0800044c  00010008  2**2
                  ALLOC
  3 .debug_info   0000078c  00000000  00000000  00010008  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 00000408  00000000  00000000  00010794  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000002dc  00000000  00000000  00010b9c  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000c0  00000000  00000000  00010e78  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   000002dd  00000000  00000000  00010f38  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    000002c5  00000000  00000000  00011215  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  000114da  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000114eb  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000204  00000000  00000000  00011520  2**2
                  CONTENTS, READONLY, DEBUGGING
```

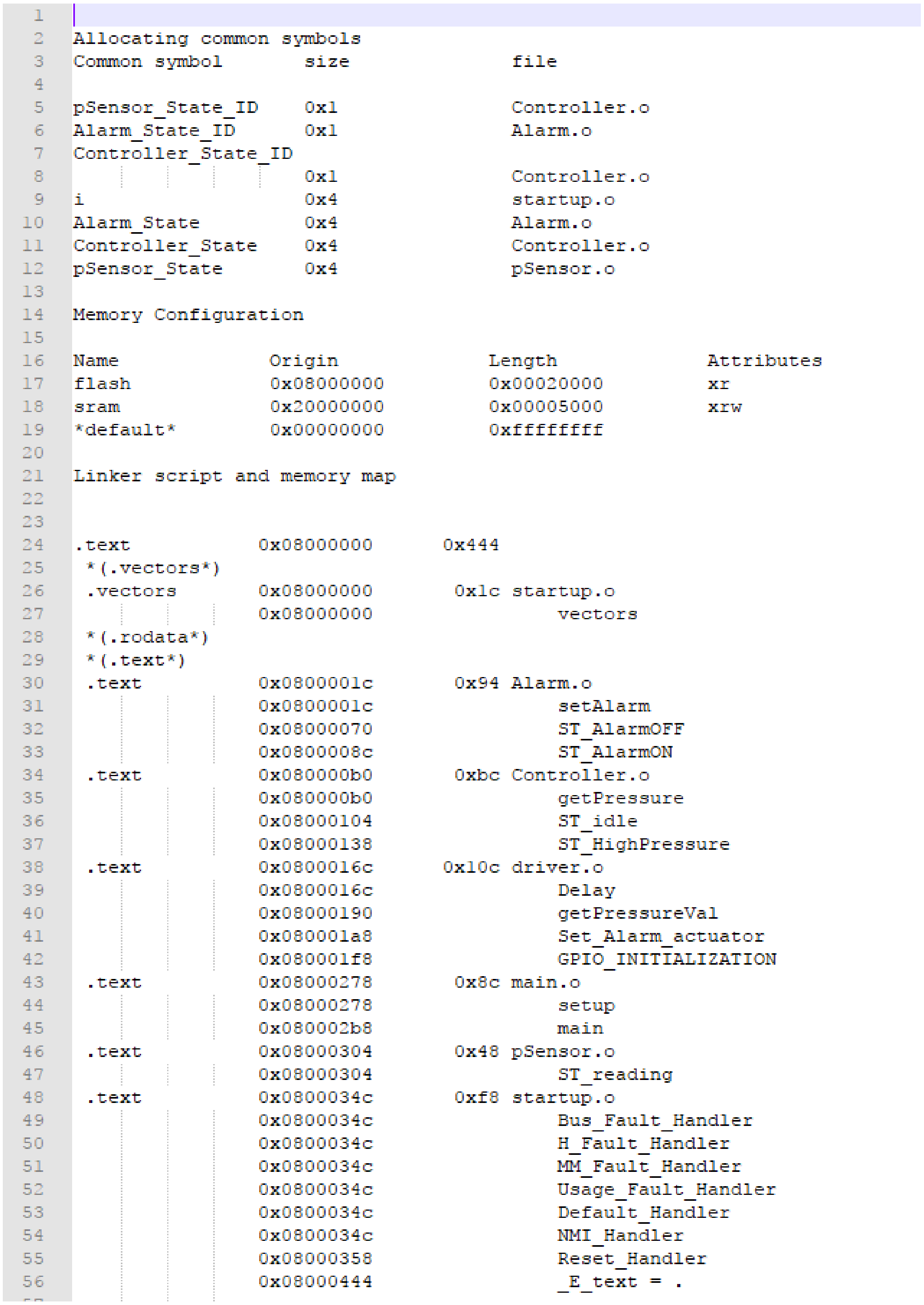
- **Symbol Table**



MINGW64:/e/EmbeddedSystemKS/Unit_5_Final_&_Project/Project_1_Pressur...

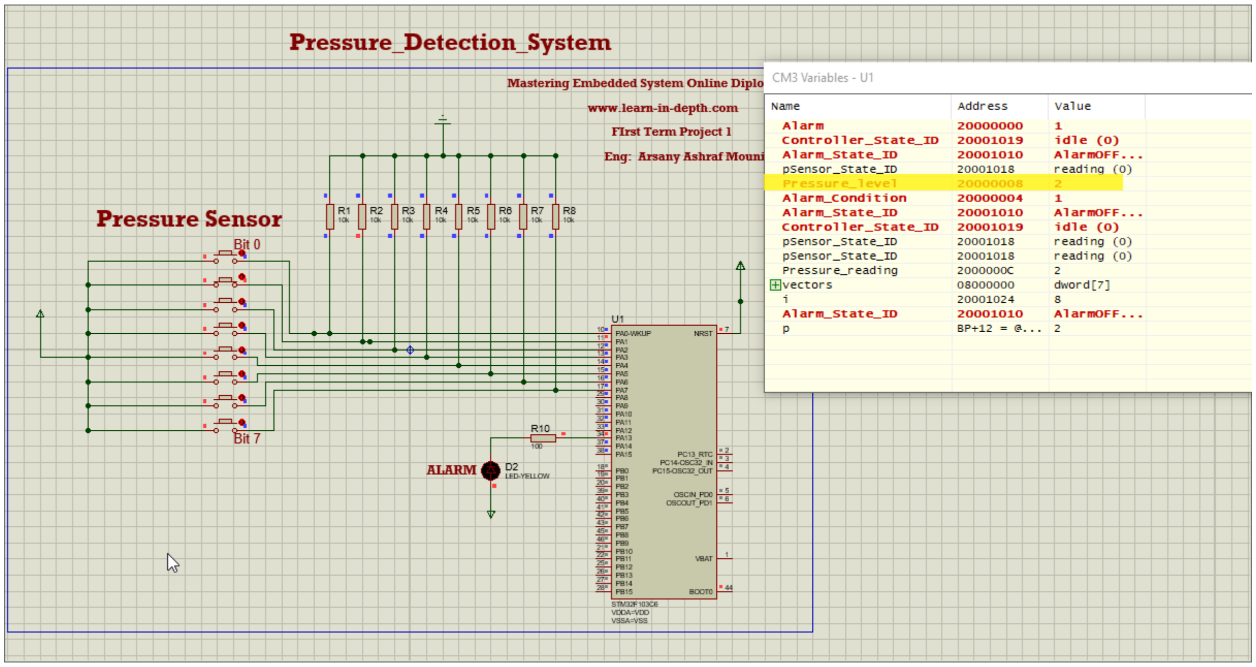
```
Arshy@Arsany MINGW64 /e/EmbeddedSystemKS/Unit_5_Final_&_Project/Project_1_Pressu
re_Sensor/FirstTerm_Project1_Pressure_Detection_System/Code
$ arm-none-eabi-nm.exe Pressure_Detection_System.elf
20000010 B _E_bss
20000008 D _E_data
08000444 T _E_text
20000008 B _S_bss
20000000 D _S_data
20001010 B _stack_top
20000000 D Alarm
20000004 D Alarm_Condition
20001014 B Alarm_State
20001010 B Alarm_State_ID
0800034c W Bus_Fault_Handler
2000101c B Controller_State
20001019 B Controller_State_ID
0800034c T Default_Handler
0800016c T Delay
080000b0 T getPressure
08000190 T getPressureVal
080001f8 T GPIO_INITIALIZATION
0800034c W H_Fault_Handler
20001024 B i
080002b8 T main
0800034c W MM_Fault_Handler
0800034c W NMI_Handler
20000008 B Pressure_level
2000000c B Pressure_reading
20001020 B pSensor_State
20001018 B pSensor_State_ID
08000358 T Reset_Handler
080001a8 T Set_Alarm_actuator
0800001c T setAlarm
08000278 T setup
08000070 T ST_AlarmOFF
0800008c T ST_AlarmON
08000138 T ST_HighPressure
08000104 T ST_idle
08000304 T ST_reading
0800034c W Usage_Fault_Handler
08000000 T vectors
```

- **Map file**



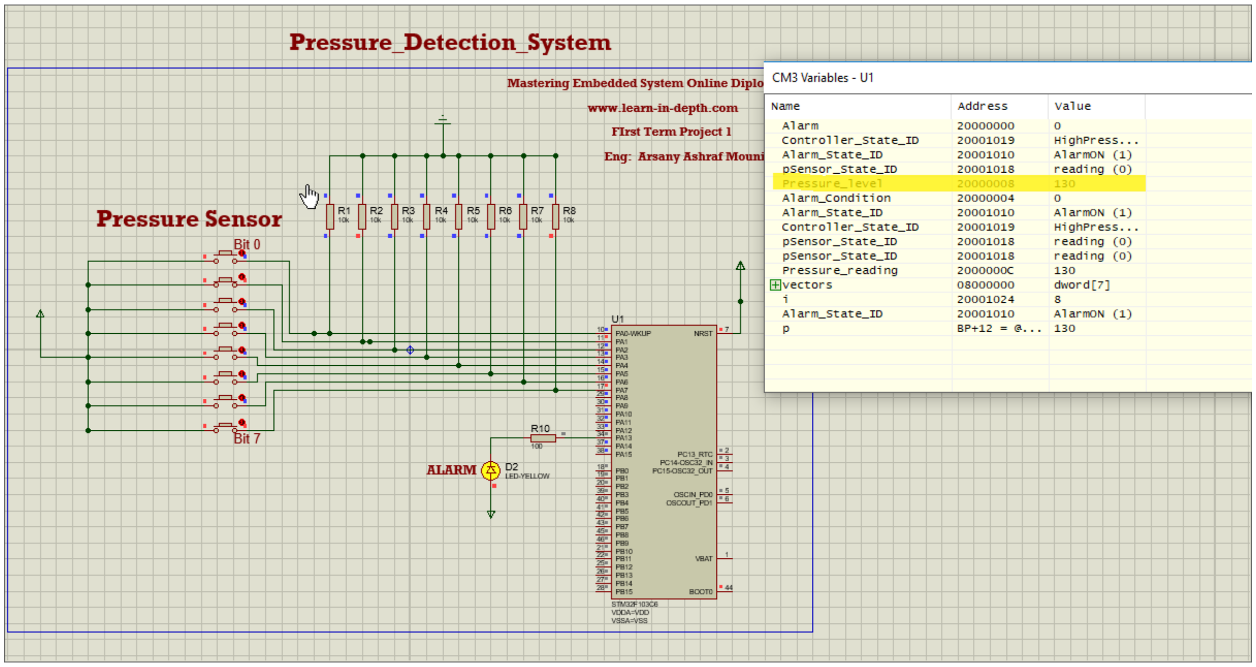
```

Allocating common symbols
Common symbol       size              file

pSensor_State_ID    0x1               Controller.o
Alarm_State_ID      0x1               Alarm.o
Controller_State_ID
                    0x1               Controller.o
i                   0x4               startup.o
Alarm_State         0x4               Alarm.o
Controller_State    0x4               Controller.o
pSensor_State       0x4               pSensor.o

Memory Configuration

Name                Origin             Length             Attributes
flash               0x08000000         0x00020000         xr
sram                0x20000000         0x00005000         xrw
*default*           0x00000000         0xffffffff

Linker script and memory map


.text               0x08000000      0x444
 *(.vectors*)
 .vectors           0x08000000       0x1c startup.o
                    0x08000000                vectors
 *(.rodata*)
 *(.text*)
 .text              0x0800001c       0x94 Alarm.o
                    0x0800001c                setAlarm
                    0x08000070                ST_AlarmOFF
                    0x0800008c                ST_AlarmON
 .text              0x080000b0       0xbc Controller.o
                    0x080000b0                getPressure
                    0x08000104                ST_idle
                    0x08000138                ST_HighPressure
 .text              0x0800016c      0x10c driver.o
                    0x0800016c                Delay
                    0x08000190                getPressureVal
                    0x080001a8                Set_Alarm_actuator
                    0x080001f8                GPIO_INITIALIZATION
 .text              0x08000278       0x8c main.o
                    0x08000278                setup
                    0x080002b8                main
 .text              0x08000304       0x48 pSensor.o
                    0x08000304                ST_reading
 .text              0x0800034c       0xf8 startup.o
                    0x0800034c                Bus_Fault_Handler
                    0x0800034c                H_Fault_Handler
                    0x0800034c                MM_Fault_Handler
                    0x0800034c                Usage_Fault_Handler
                    0x0800034c                Default_Handler
                    0x0800034c                NMI_Handler
                    0x08000358                Reset_Handler
                    0x08000444                _E_text = .
```

Simulation:-



CM3 Variables - U1

Name	Address	Value
Alarm	20000000	1
Controller_State_ID	20001019	idle (0)
Alarm_State_ID	20001010	AlarmOFF...
pSensor_State_ID	20001018	reading (0)
Pressure_level	20000008	2
Alarm_Condition	20000004	1
Alarm_State_ID	20001010	AlarmOFF...
Controller_State_ID	20001019	idle (0)
pSensor_State_ID	20001018	reading (0)
pSensor_State_ID	20001018	reading (0)
Pressure_reading	2000000C	2
vectors	08000000	dword[7]
i	20001024	8
Alarm_State_ID	20001010	AlarmOFF...
p	BP+12 = @...	2



CM3 Variables - U1

Name	Address	Value
Alarm	20000000	0
Controller_State_ID	20001019	HighPress...
Alarm_State_ID	20001010	AlarmON (1)
pSensor_State_ID	20001018	reading (0)
Pressure_level	20000008	130
Alarm_Condition	20000004	0
Alarm_State_ID	20001010	AlarmON (1)
Controller_State_ID	20001019	HighPress...
pSensor_State_ID	20001018	reading (0)
pSensor_State_ID	20001018	reading (0)
Pressure_reading	2000000C	130
vectors	08000000	dword[7]
i	20001024	8
Alarm_State_ID	20001010	AlarmON (1)
p	BP+12 = @...	130