Mastering Embedded System Online Diploma

www.learn-in-depth.com

First Term (Final Project 1)

Eng.Mostafa Hamed Besher

My Profile:

www.learn-in-depth.com/online-diploma/mostafahamed241@gmail.com

Project 1

Pressure Detection System

- **Abstract :**

  This Project Function Is To Detect High Pressure in a Plane Cabin.

  If High Pressure Detected , It Raises Alarm Which Is Turnning On Led For 60 Seconds.

- **Project Design :**

  To Efficiently Design This System I Go Through These Design Stages

  - **Case Study**
  - **Design Method**
  - **Requirements**
  - **Space Exploration/Partioning**
  - **System Analysis**
  - **System Design**

- **Case Study :**

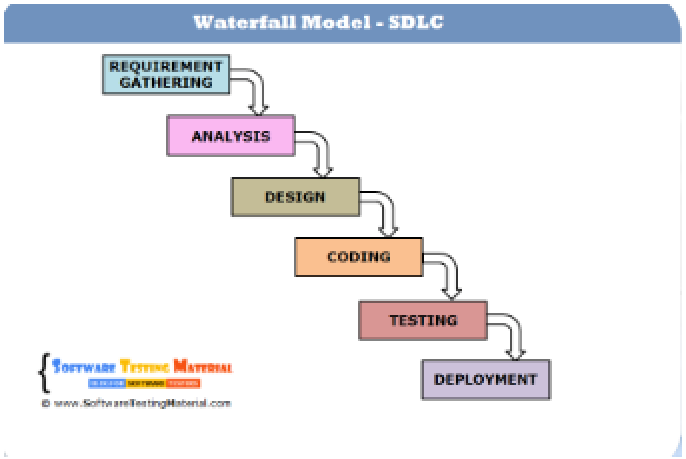
  A Client Expects To Deliver The Software Of The Following System :

  - **Specifications :**
    - **Detection System which Informs The Crew Of a Cabin With An Alarm When Pressure Exceeding 20 Bars In The Cabin**
    - **The Alarm Duration Equals 60 Seconds**
    - **Keep Track Of The Measured Values**

- 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
    - Versioning The ”keep Track Of Measured Value” Option is not Modeled in The First Version Of The Design

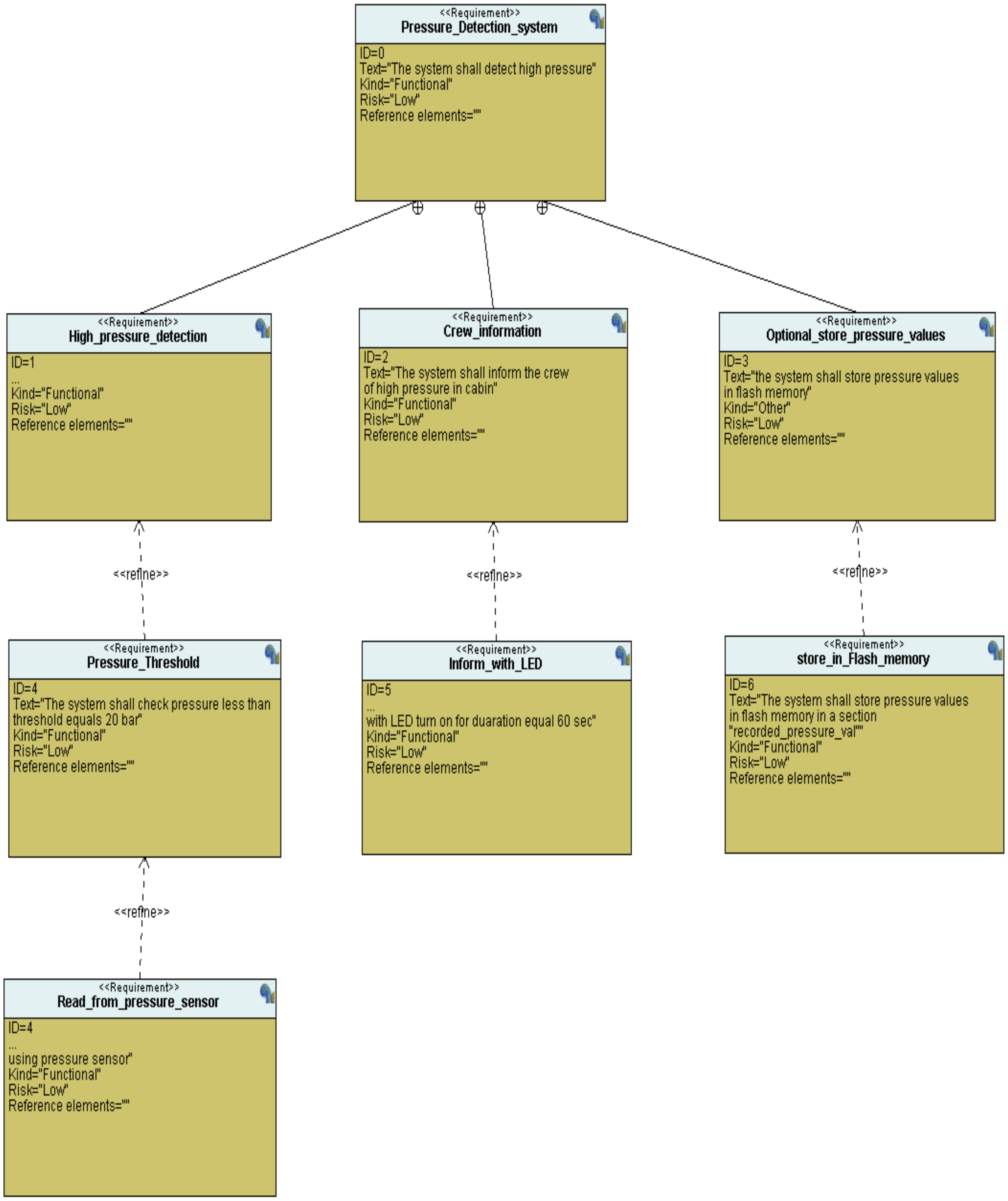
❖Design Method :

I used Water Fall Design Method



❖Requirements :

- **Requirement Diagram**



<<Requirement>>
Pressure_Detection_system
ID=0
Text="The system shall detect high pressure"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
High_pressure_detection
ID=1
...
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Crew_information
ID=2
Text="The system shall inform the crew of high pressure in cabin"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Optional_store_pressure_values
ID=3
Text="the system shall store pressure values in flash memory"
Kind="Other"
Risk="Low"
Reference elements=""

<<Requirement>>
Pressure_Threshold
ID=4
Text="The system shall check pressure less than threshold equals 20 bar"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Inform_with_LED
ID=5
...
with LED turn on for duaration equal 60 sec"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
store_in_Flash_memory
ID=6
Text="The system shall store pressure values in flash memory in a section "recorded_pressure_val""
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Read_from_pressure_sensor
ID=4
...
using pressure sensor"
Kind="Functional"
Risk="Low"
Reference elements=""

❖Space Exploration/Partioning

- **I Used STM32F103C6 SOC Which Based On Arm Cortex–M3 Micro Processor Which Specifications Are :**

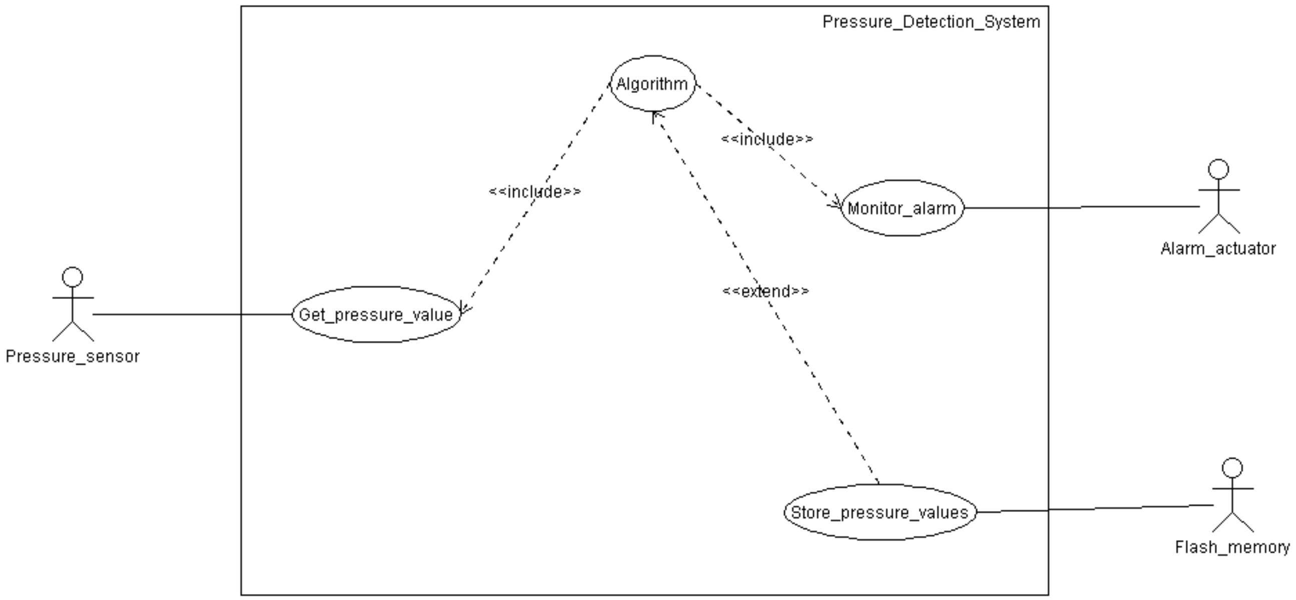
  - ARM 32-bit Cortex™-M3 CPU Core
    72 MHz maximum frequency,1.25 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state memory access

  - Single-cycle multiplication and hardware division

- **Hw/Sw partitioning can speedup software**
- **Can reduce energy too**
- **Can reduce cost**

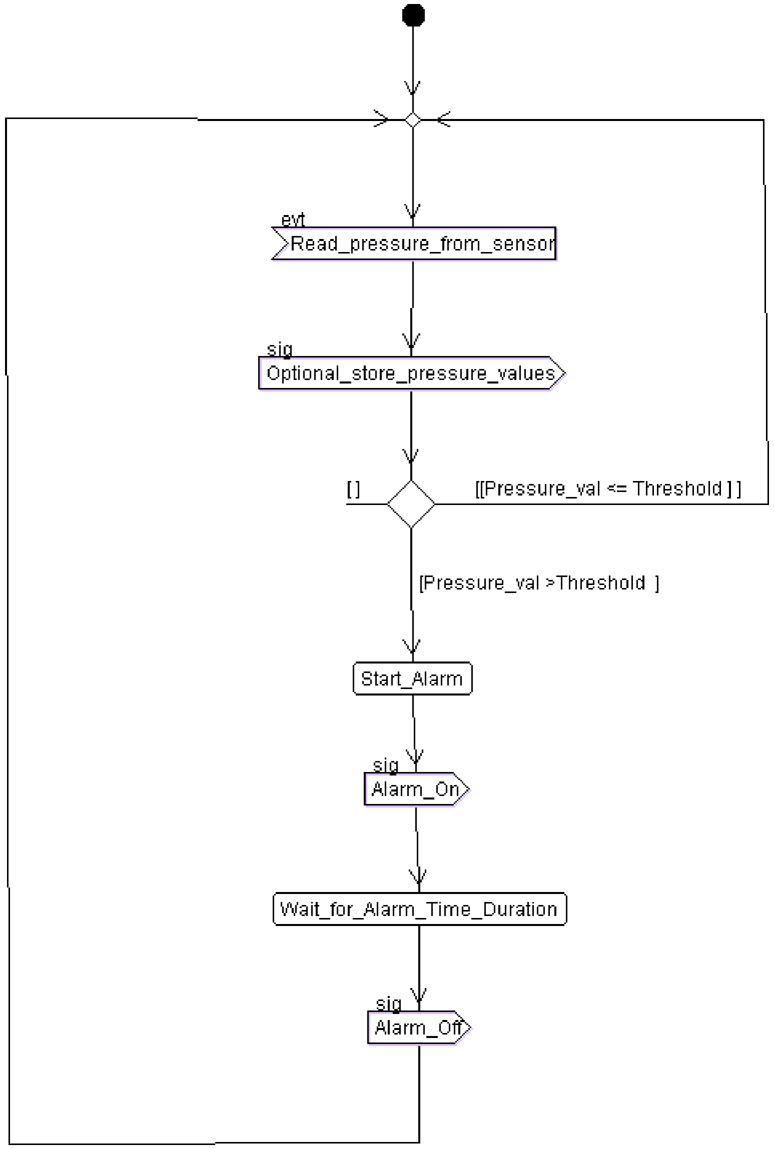
❖System Analysis

- **Use Case Diagram**



Use Case Diagram function is to inform the client
Of what the system main functions and define system
Boundary With some level of abstraction of system
details

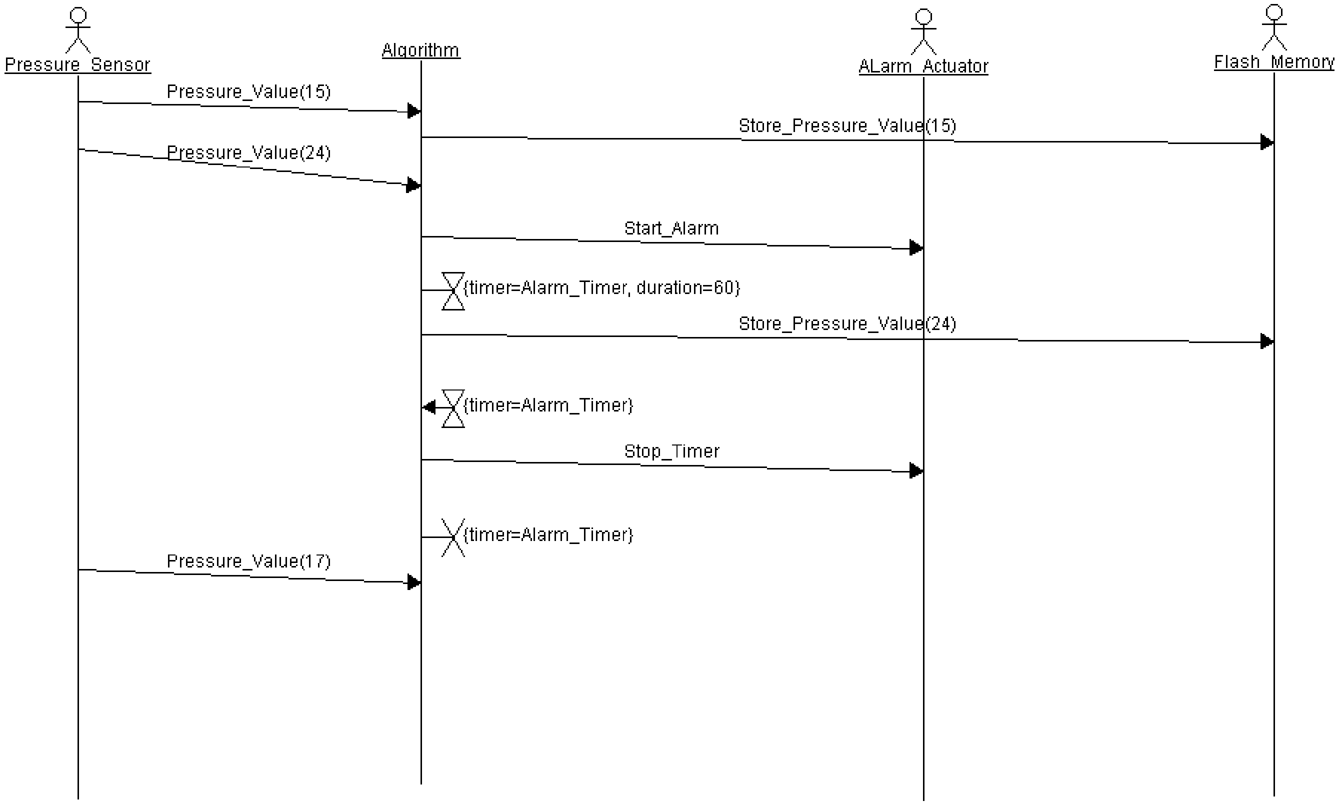
- **Activity Diagram**



evt
Read_pressure_from_sensor
sig
Optional_store_pressure_values
[]
[[Pressure_val <= Threshold]]
[Pressure_val >Threshold]
Start_Alarm
sig
Alarm_On
Wait_for_Alarm_Time_Duration
sig
Alarm_Off

Activity function is to show the relations Between main functions of the system And describe the work flow of the system

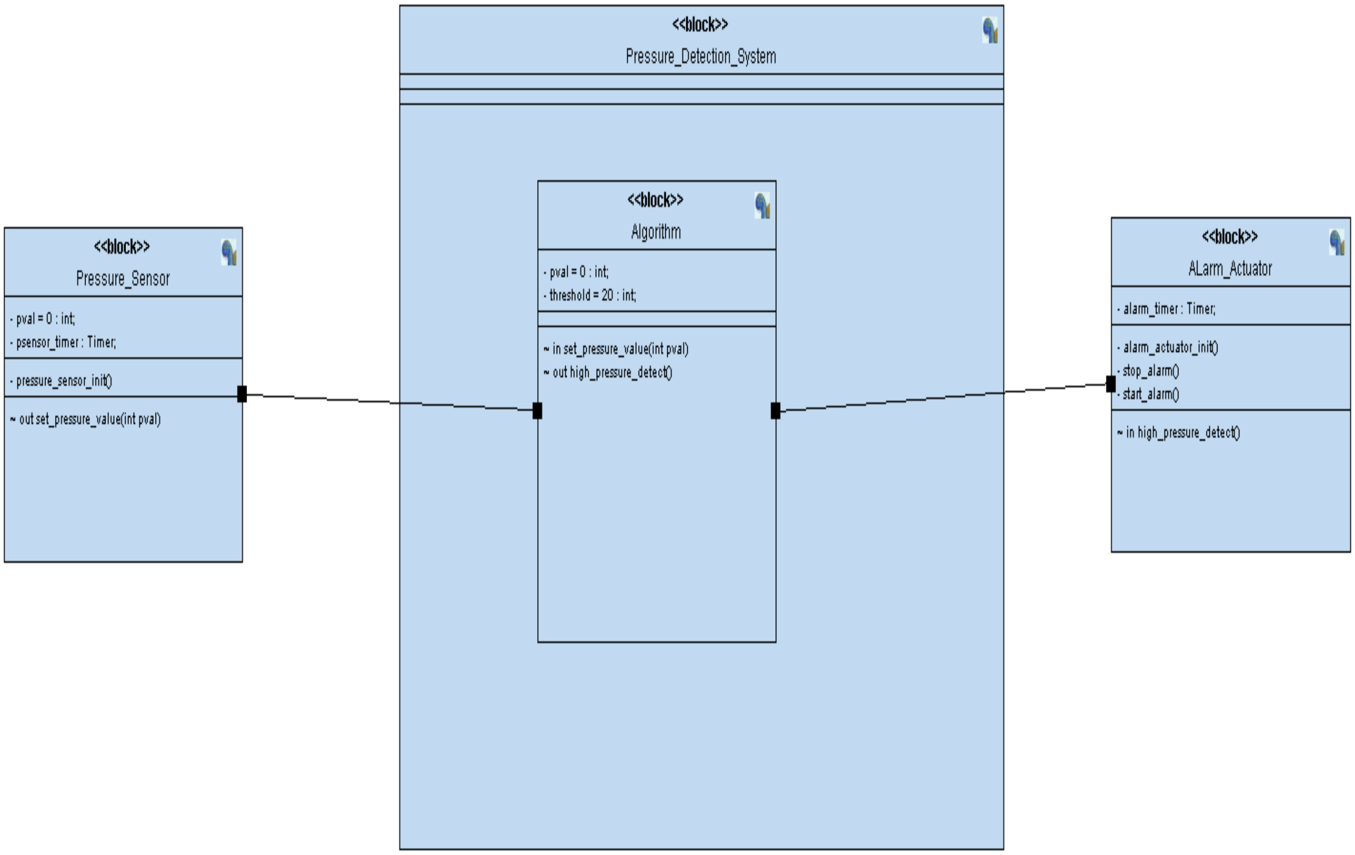
- **Sequence Diagram**



- ○ **Sequence Diagram is an interaction diagram that details how operations are carried out and Shows What messages are sent and when.**
- ○ **Sequence diagrams are organized according to Time.**

❖System Design

- **Block Diagram**



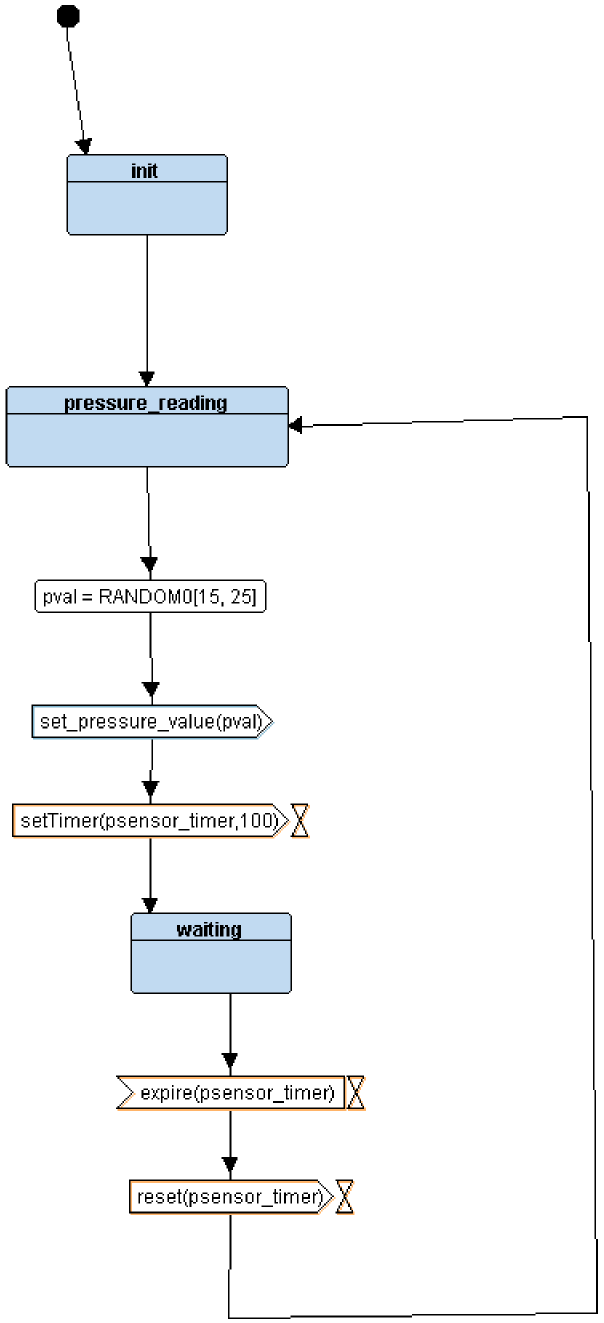
I used here multiple modules in block diagram, one module For pressure sensor, one for alarm actuator and the last one For main algorithm of the system which controls it.

- **State Machine Diagram**

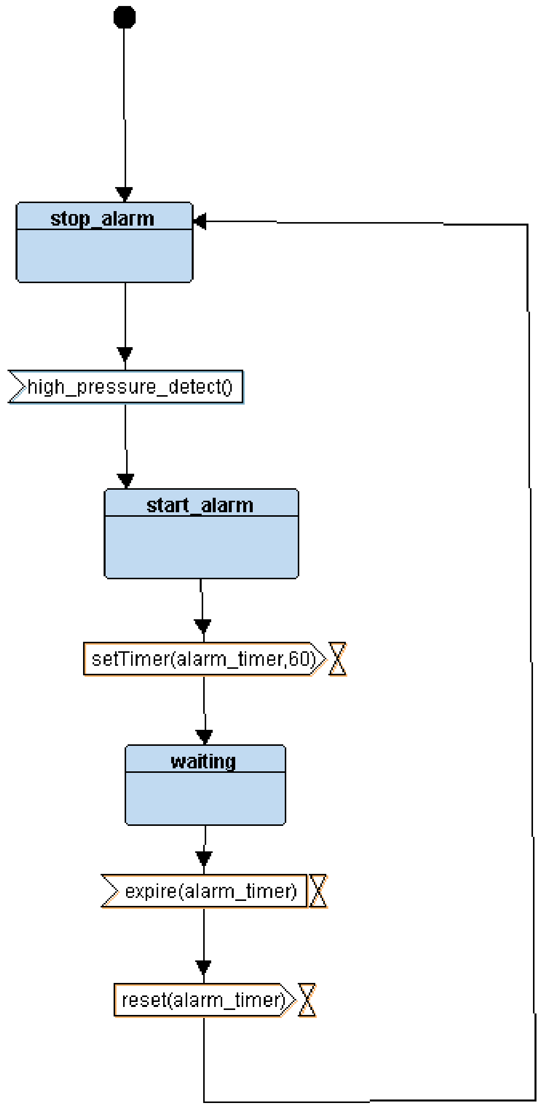
This diagram describes all states for each module

And how it can switchs between different states.

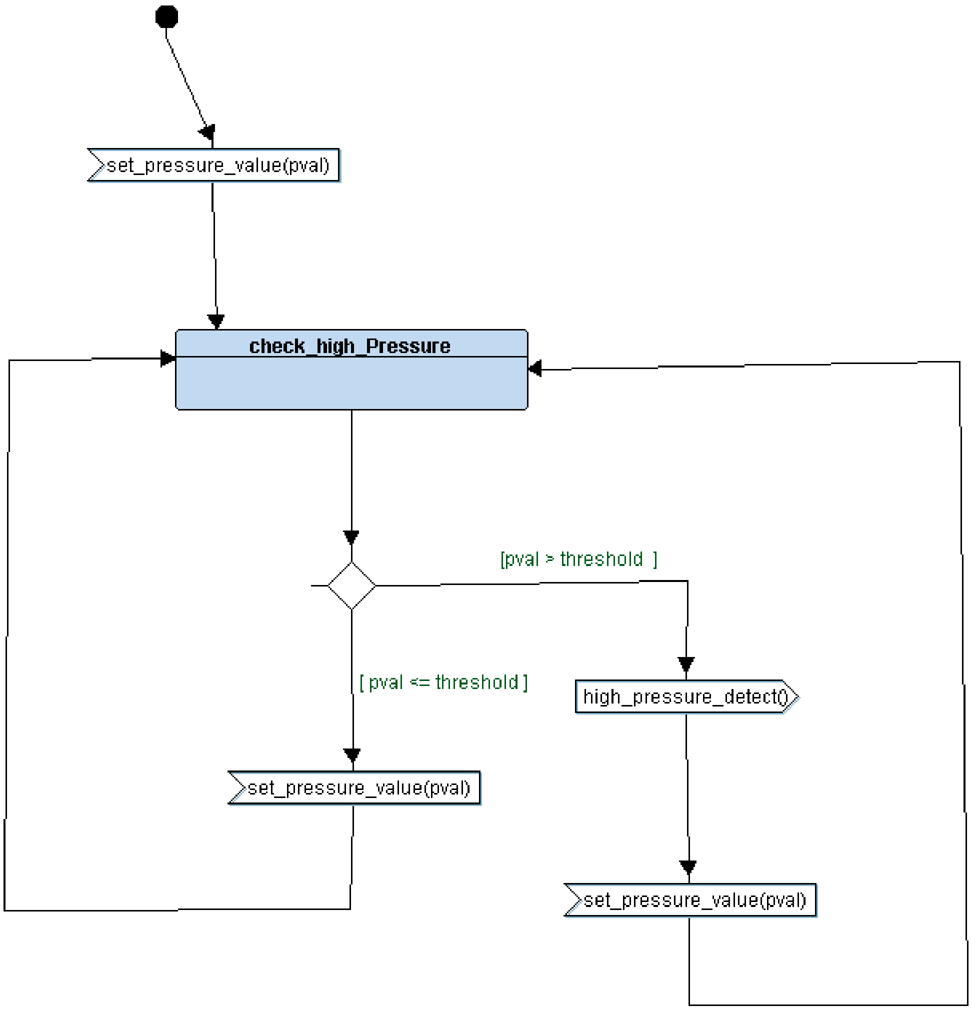
  - **For pressure sensor module**



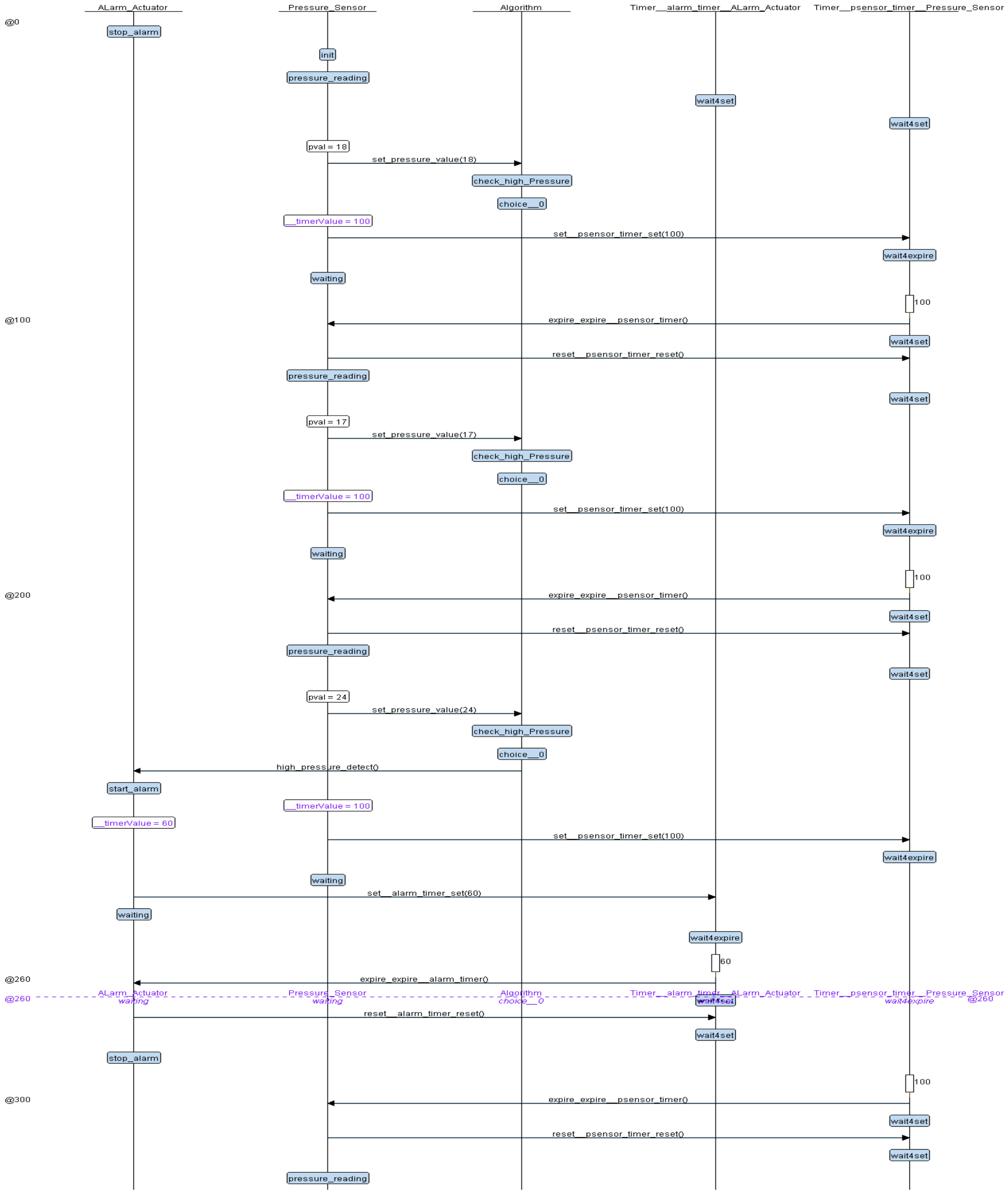
- **For alarm actuator module**



- **For main algorithm Module**



- **Simulation**


ALarm_Actuator
Pressure_Sensor
Algorithm
Timer__alarm_timer__ALarm_Actuator
Timer__psensor_timer__Pressure_Sensor
@0
stop_alarm
init
pressure_reading
wait4set
wait4set
pval = 18
set_pressure_value(18)
check_high_Pressure
choice__0
__timerValue = 100
set__psensor_timer_set(100)
wait4expire
waiting
100
@100
expire_expire__psensor_timer()
wait4set
reset__psensor_timer_reset()
pressure_reading
wait4set
pval = 17
set_pressure_value(17)
check_high_Pressure
choice__0
__timerValue = 100
set__psensor_timer_set(100)
wait4expire
waiting
100
@200
expire_expire__psensor_timer()
wait4set
reset__psensor_timer_reset()
pressure_reading
wait4set
pval = 24
set_pressure_value(24)
check_high_Pressure
choice__0
high_pressure_detect()
start_alarm
__timerValue = 100
__timerValue = 60
set__psensor_timer_set(100)
wait4expire
waiting
set__alarm_timer_set(60)
waiting
wait4expire
60
@260
expire_expire__alarm_timer()
@260
ALarm_Actuator
waiting
Pressure_Sensor
waiting
Algorithm
choice__0
Timer__alarm_timer__ALarm_Actuator
wait4set
Timer__psensor_timer__Pressure_Sensor
wait4expire
@260
reset__alarm_timer_reset()
wait4set
stop_alarm
100
@300
expire_expire__psensor_timer()
wait4set
reset__psensor_timer_reset()
wait4set
pressure_reading

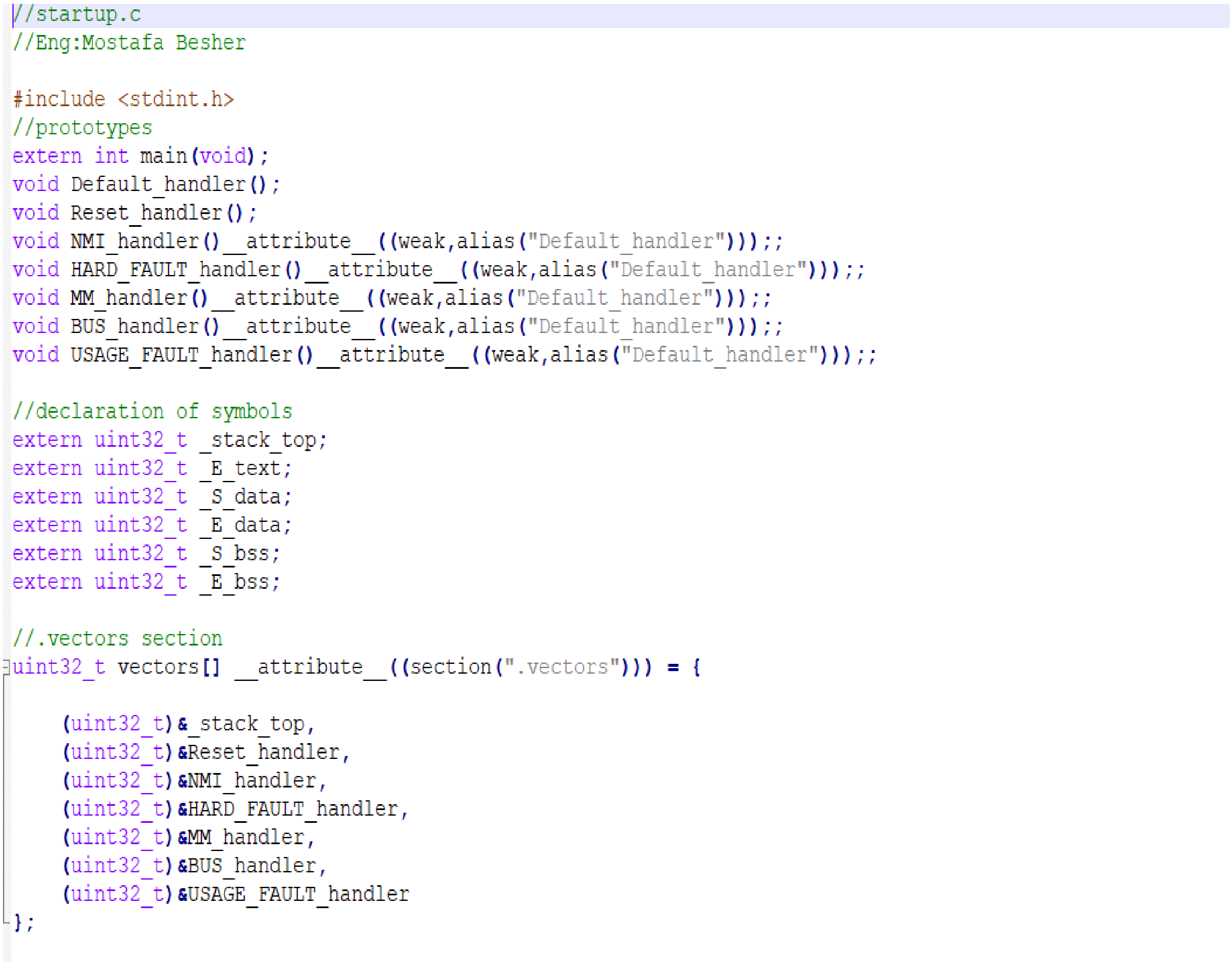
➢ Project Implementation :

❖Writing code

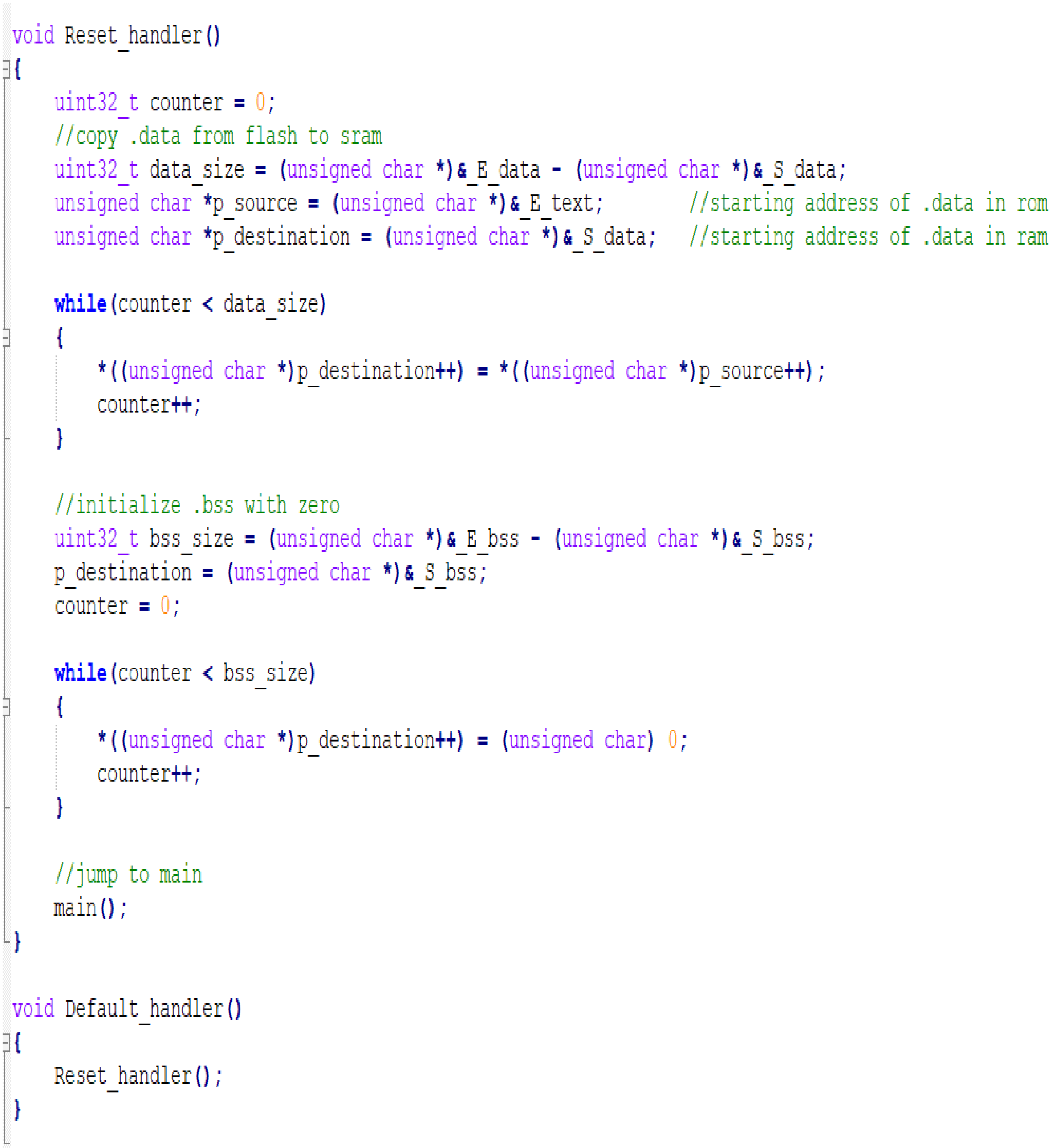
You can Find All Project Codes in My Github Repository

https://github.com/mostafahamedbesher/Embedded_Online_Diploma

- **Startup.c**

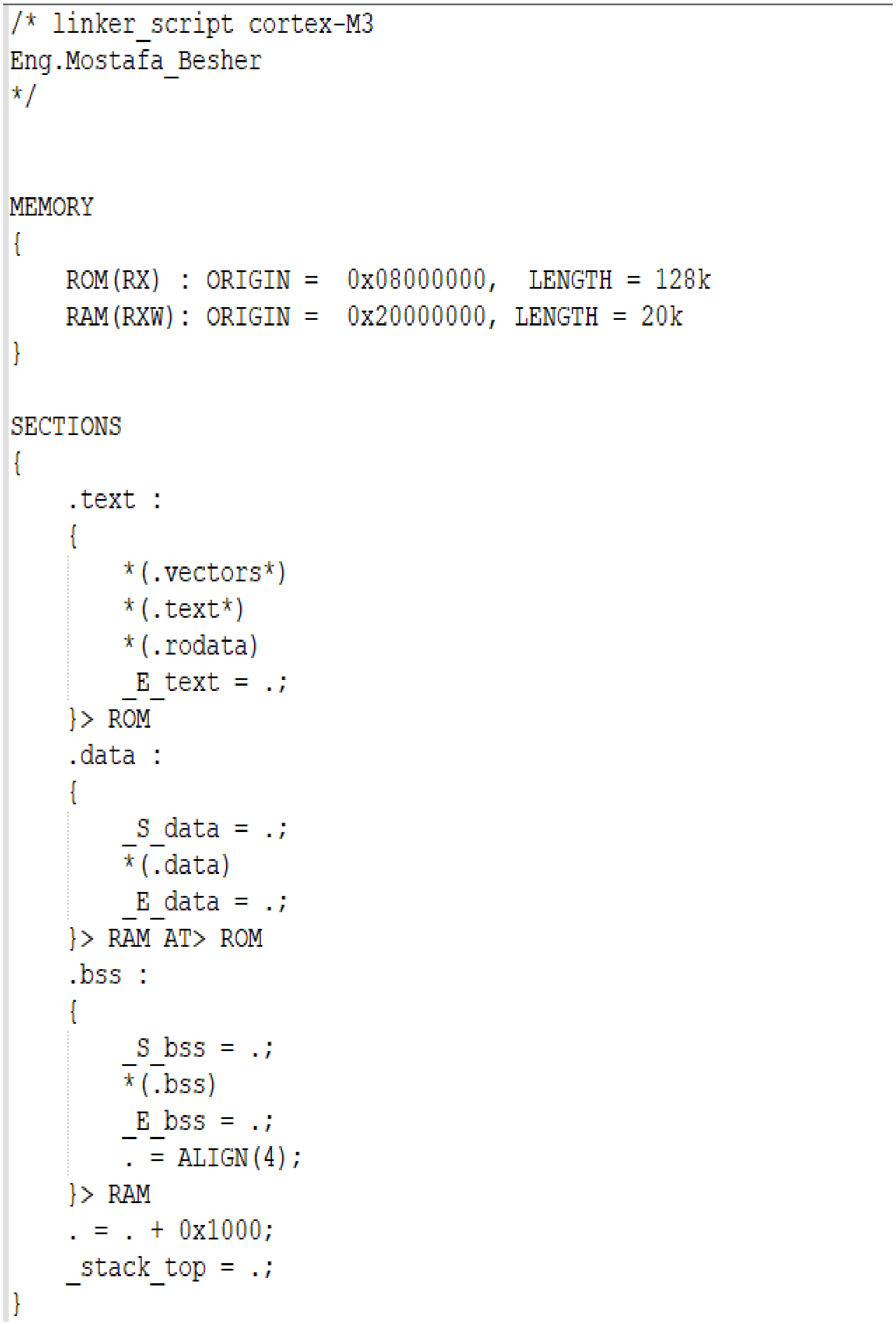


```c
//startup.c
//Eng:Mostafa Besher

#include <stdint.h>
//prototypes
extern int main(void);
void Default_handler();
void Reset_handler();
void NMI_handler()__attribute__((weak,alias("Default_handler")));;
void HARD_FAULT_handler()__attribute__((weak,alias("Default_handler")));;
void MM_handler()__attribute__((weak,alias("Default_handler")));;
void BUS_handler()__attribute__((weak,alias("Default_handler")));;
void USAGE_FAULT_handler()__attribute__((weak,alias("Default_handler")));;

//declaration of symbols
extern uint32_t _stack_top;
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;

//.vectors section
uint32_t vectors[] __attribute__((section(".vectors"))) = {

    (uint32_t)&_stack_top,
    (uint32_t)&Reset_handler,
    (uint32_t)&NMI_handler,
    (uint32_t)&HARD_FAULT_handler,
    (uint32_t)&MM_handler,
    (uint32_t)&BUS_handler,
    (uint32_t)&USAGE_FAULT_handler
};
```



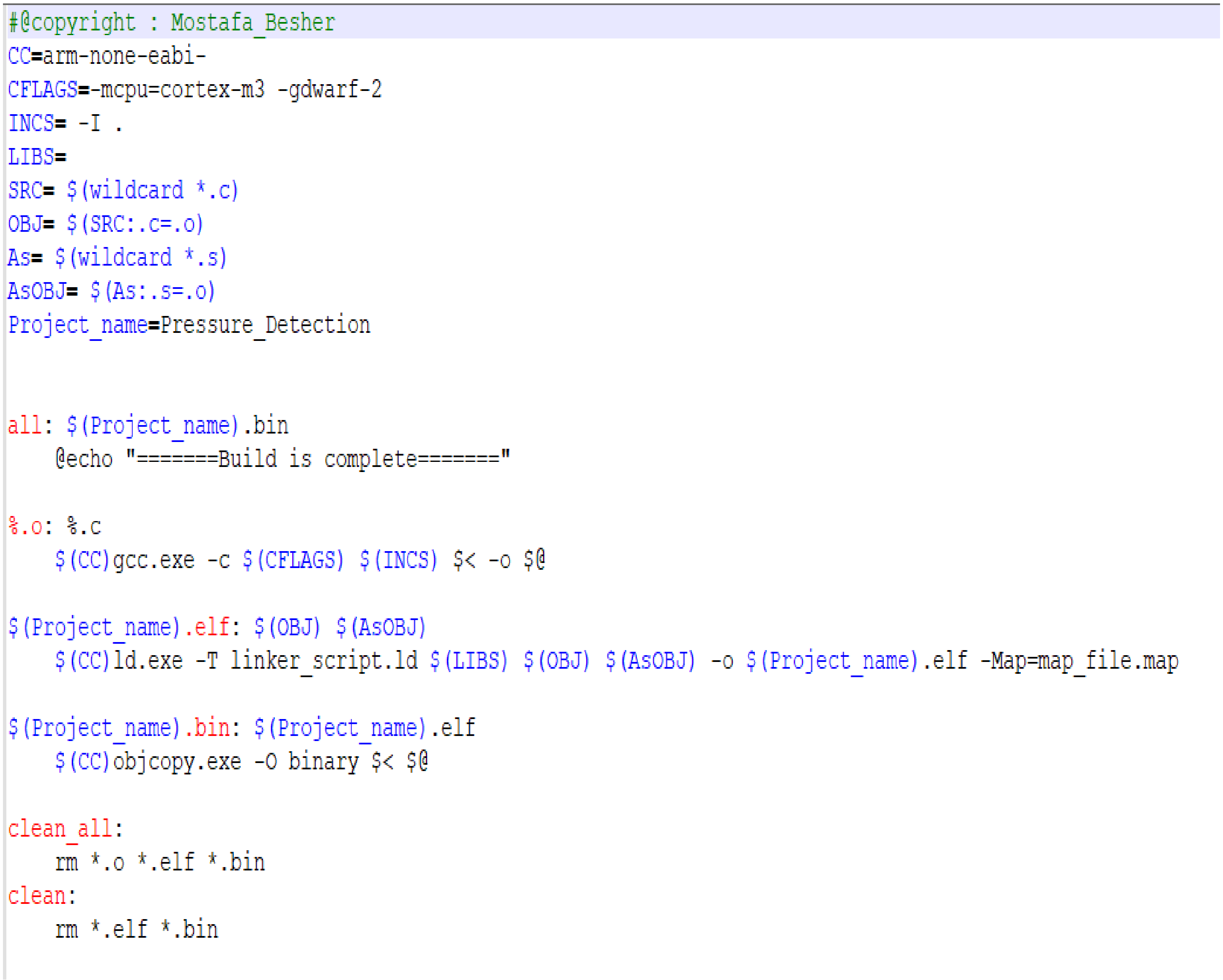
```
void Reset_handler()
{
    uint32_t counter = 0;
    //copy .data from flash to sram
    uint32_t data_size = (unsigned char *)&_E_data - (unsigned char *)&_S_data;
    unsigned char *p_source = (unsigned char *)&_E_text;        //starting address of .data in rom
    unsigned char *p_destination = (unsigned char *)&_S_data;   //starting address of .data in ram

    while(counter < data_size)
    {
        *((unsigned char *)p_destination++) = *((unsigned char *)p_source++);
        counter++;
    }

    //initialize .bss with zero
    uint32_t bss_size = (unsigned char *)&_E_bss - (unsigned char *)&_S_bss;
    p_destination = (unsigned char *)&_S_bss;
    counter = 0;

    while(counter < bss_size)
    {
        *((unsigned char *)p_destination++) = (unsigned char) 0;
        counter++;
    }

    //jump to main
    main();
}

void Default_handler()
{
    Reset_handler();
}
```

- **Linker_script**



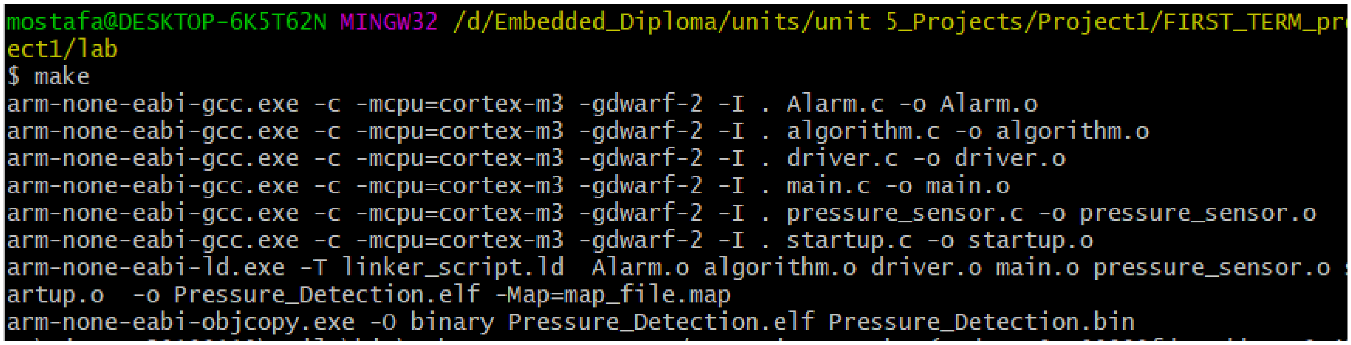
```
/* linker_script cortex-M3
Eng.Mostafa_Besher
*/


MEMORY
{
    ROM(RX) : ORIGIN =  0x08000000,  LENGTH = 128k
    RAM(RXW): ORIGIN =  0x20000000, LENGTH = 20k
}

SECTIONS
{
    .text :
    {
        *(.vectors*)
        *(.text*)
        *(.rodata)
        _E_text = .;
    }> ROM
    .data :
    {
        _S_data = .;
        *(.data)
        _E_data = .;
    }> RAM AT> ROM
    .bss :
    {
        _S_bss = .;
        *(.bss)
        _E_bss = .;
        . = ALIGN(4);
    }> RAM
    . = . + 0x1000;
    _stack_top = .;
}
```

- ## Makefile



```
#@copyright : Mostafa_Besher
CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -gdwarf-2
INCS= -I .
LIBS=
SRC= $(wildcard *.c)
OBJ= $(SRC:.c=.o)
As= $(wildcard *.s)
AsOBJ= $(As:.s=.o)
Project_name=Pressure_Detection


all: $(Project_name).bin
    @echo "=======Build is complete======="

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

$(Project_name).elf: $(OBJ) $(AsOBJ)
    $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $(Project_name).elf -Map=map_file.map

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

clean_all:
    rm *.o *.elf *.bin
clean:
    rm *.elf *.bin
```

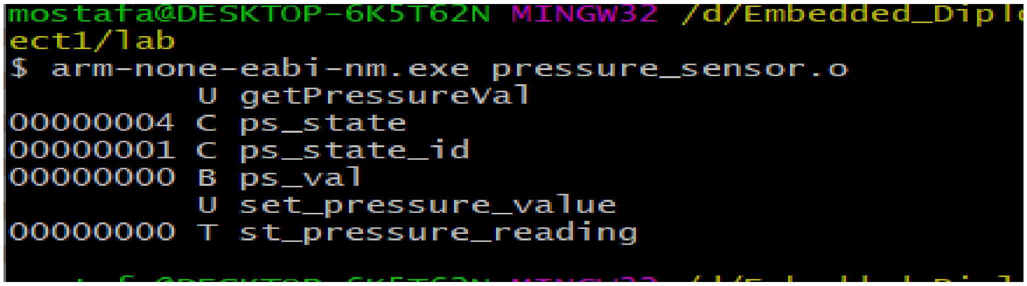
❖Get Object Files Using Makefile :



```
mostafa@DESKTOP-6K5T62N MINGW32 /d/Embedded_Diploma/units/unit 5_Projects/Project1/FIRST_TERM_pr
ect1/lab
$ make
arm-none-eabi-gcc.exe -c -mcpu=cortex-m3 -gdwarf-2 -I . Alarm.c -o Alarm.o
arm-none-eabi-gcc.exe -c -mcpu=cortex-m3 -gdwarf-2 -I . algorithm.c -o algorithm.o
arm-none-eabi-gcc.exe -c -mcpu=cortex-m3 -gdwarf-2 -I . driver.c -o driver.o
arm-none-eabi-gcc.exe -c -mcpu=cortex-m3 -gdwarf-2 -I . main.c -o main.o
arm-none-eabi-gcc.exe -c -mcpu=cortex-m3 -gdwarf-2 -I . pressure_sensor.c -o pressure_sensor.o
arm-none-eabi-gcc.exe -c -mcpu=cortex-m3 -gdwarf-2 -I . startup.c -o startup.o
arm-none-eabi-ld.exe -T linker_script.ld  Alarm.o algorithm.o driver.o main.o pressure_sensor.o s
artup.o  -o Pressure_Detection.elf -Map=map_file.map
arm-none-eabi-objcopy.exe -O binary Pressure_Detection.elf Pressure_Detection.bin
```

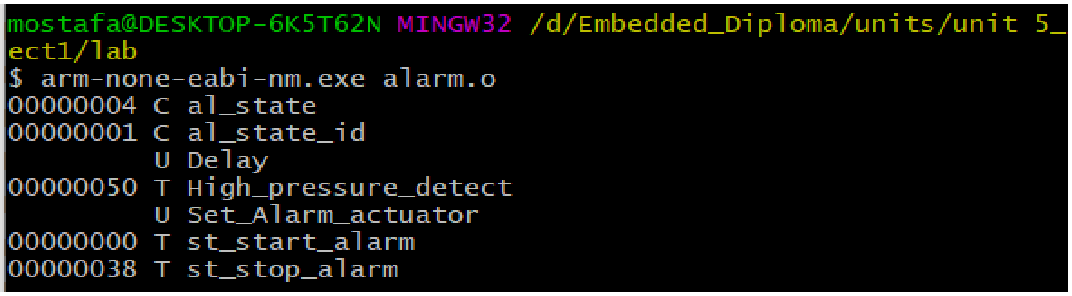
❖Show Symbols For:

- **Pressure_Sensor.o**



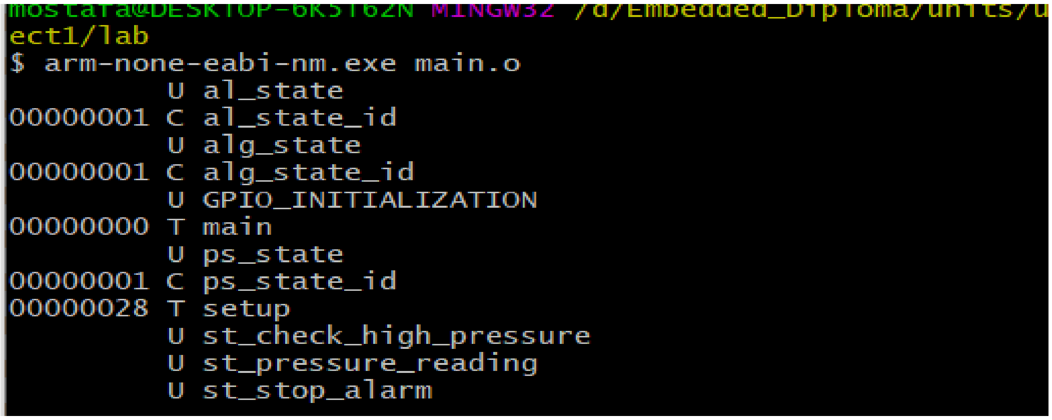
```
mostafa@DESKTOP-6K5T62N MINGW32 /d/Embedded_Dipl
ect1/lab
$ arm-none-eabi-nm.exe pressure_sensor.o
         U getPressureVal
00000004 C ps_state
00000001 C ps_state_id
00000000 B ps_val
         U set_pressure_value
00000000 T st_pressure_reading
```

- **Alarm.o**



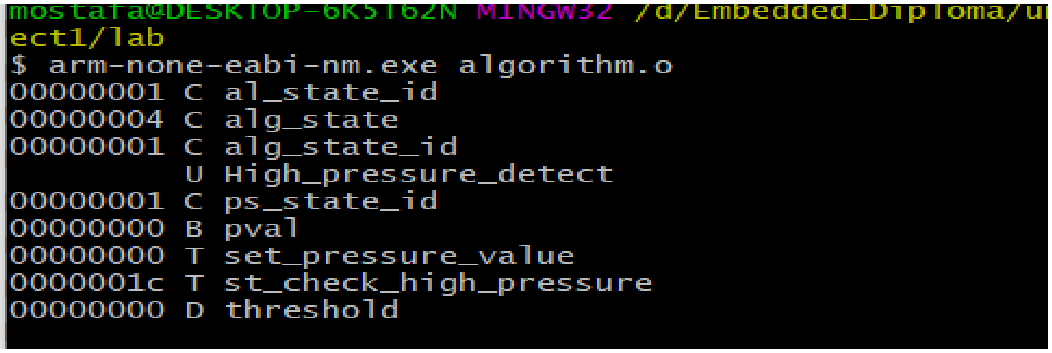
```
mostafa@DESKTOP-6K5T62N MINGW32 /d/Embedded_Diploma/units/unit 5_
ect1/lab
$ arm-none-eabi-nm.exe alarm.o
00000004 C al_state
00000001 C al_state_id
         U Delay
00000050 T High_pressure_detect
         U Set_Alarm_actuator
00000000 T st_start_alarm
00000038 T st_stop_alarm
```

- **Main.o**



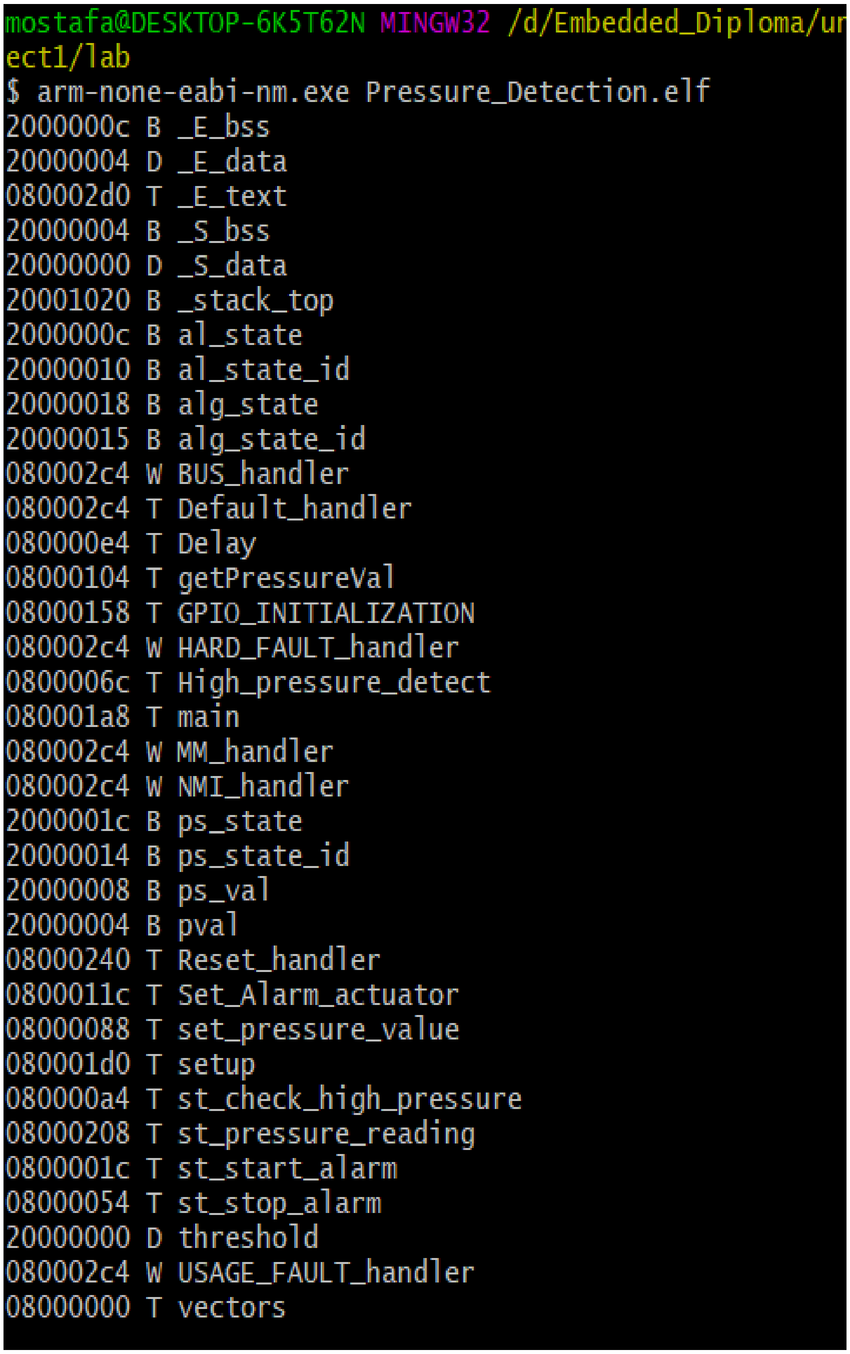
```
mostafa@DESKTOP-6K5T62N MINGW32 /d/Embedded_Diploma/units/u
ect1/lab
$ arm-none-eabi-nm.exe main.o
         U al_state
00000001 C al_state_id
         U alg_state
00000001 C alg_state_id
         U GPIO_INITIALIZATION
00000000 T main
         U ps_state
00000001 C ps_state_id
00000028 T setup
         U st_check_high_pressure
         U st_pressure_reading
         U st_stop_alarm
```

- **Algorithm.o**



```
mostafa@DESKTOP-6K5T62N MINGW32 /d/Embedded_Diploma/u
ect1/lab
$ arm-none-eabi-nm.exe algorithm.o
00000001 C al_state_id
00000004 C alg_state
00000001 C alg_state_id
         U High_pressure_detect
00000001 C ps_state_id
00000000 B pval
00000000 T set_pressure_value
0000001c T st_check_high_pressure
00000000 D threshold
```

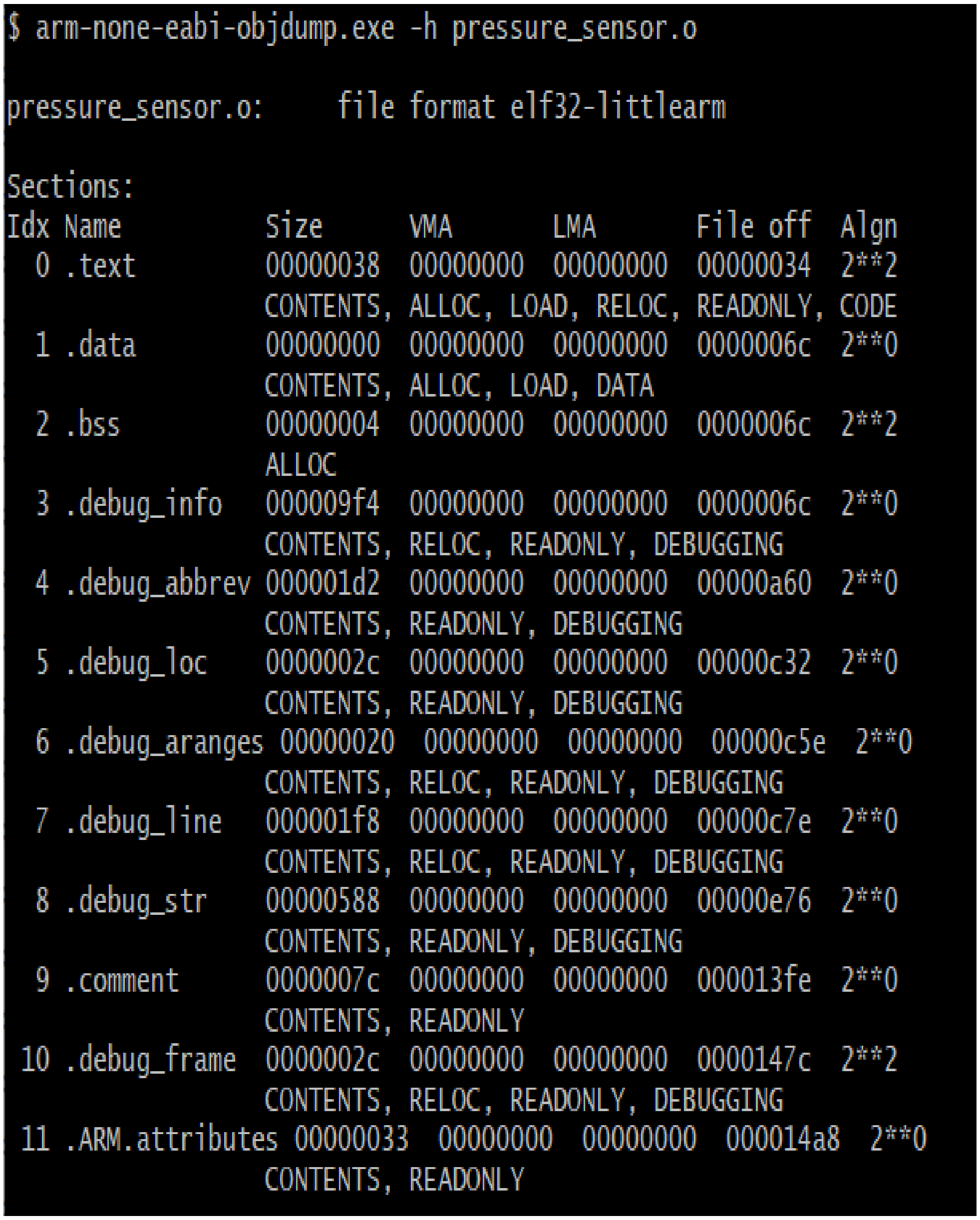
- **Pressure_Detection.elf**



```
mostafa@DESKTOP-6K5T62N MINGW32 /d/Embedded_Diploma/ur
ect1/lab
$ arm-none-eabi-nm.exe Pressure_Detection.elf
2000000c B _E_bss
20000004 D _E_data
080002d0 T _E_text
20000004 B _S_bss
20000000 D _S_data
20001020 B _stack_top
2000000c B al_state
20000010 B al_state_id
20000018 B alg_state
20000015 B alg_state_id
080002c4 W BUS_handler
080002c4 T Default_handler
080000e4 T Delay
08000104 T getPressureVal
08000158 T GPIO_INITIALIZATION
080002c4 W HARD_FAULT_handler
0800006c T High_pressure_detect
080001a8 T main
080002c4 W MM_handler
080002c4 W NMI_handler
2000001c B ps_state
20000014 B ps_state_id
20000008 B ps_val
20000004 B pval
08000240 T Reset_handler
0800011c T Set_Alarm_actuator
08000088 T set_pressure_value
080001d0 T setup
080000a4 T st_check_high_pressure
08000208 T st_pressure_reading
0800001c T st_start_alarm
08000054 T st_stop_alarm
20000000 D threshold
080002c4 W USAGE_FAULT_handler
08000000 T vectors
```

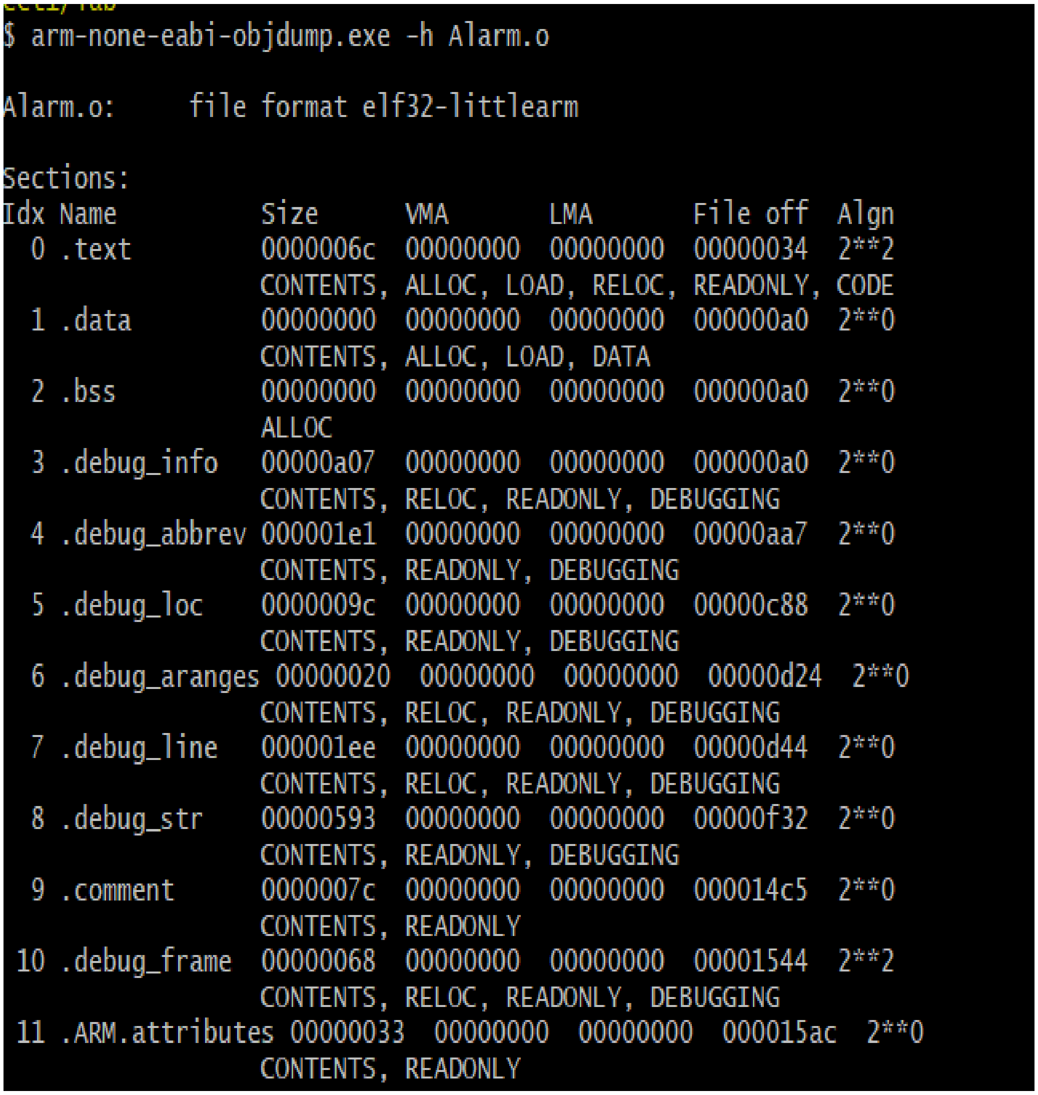
❖Show Sections For :

- **Pressure_Sensor.o**



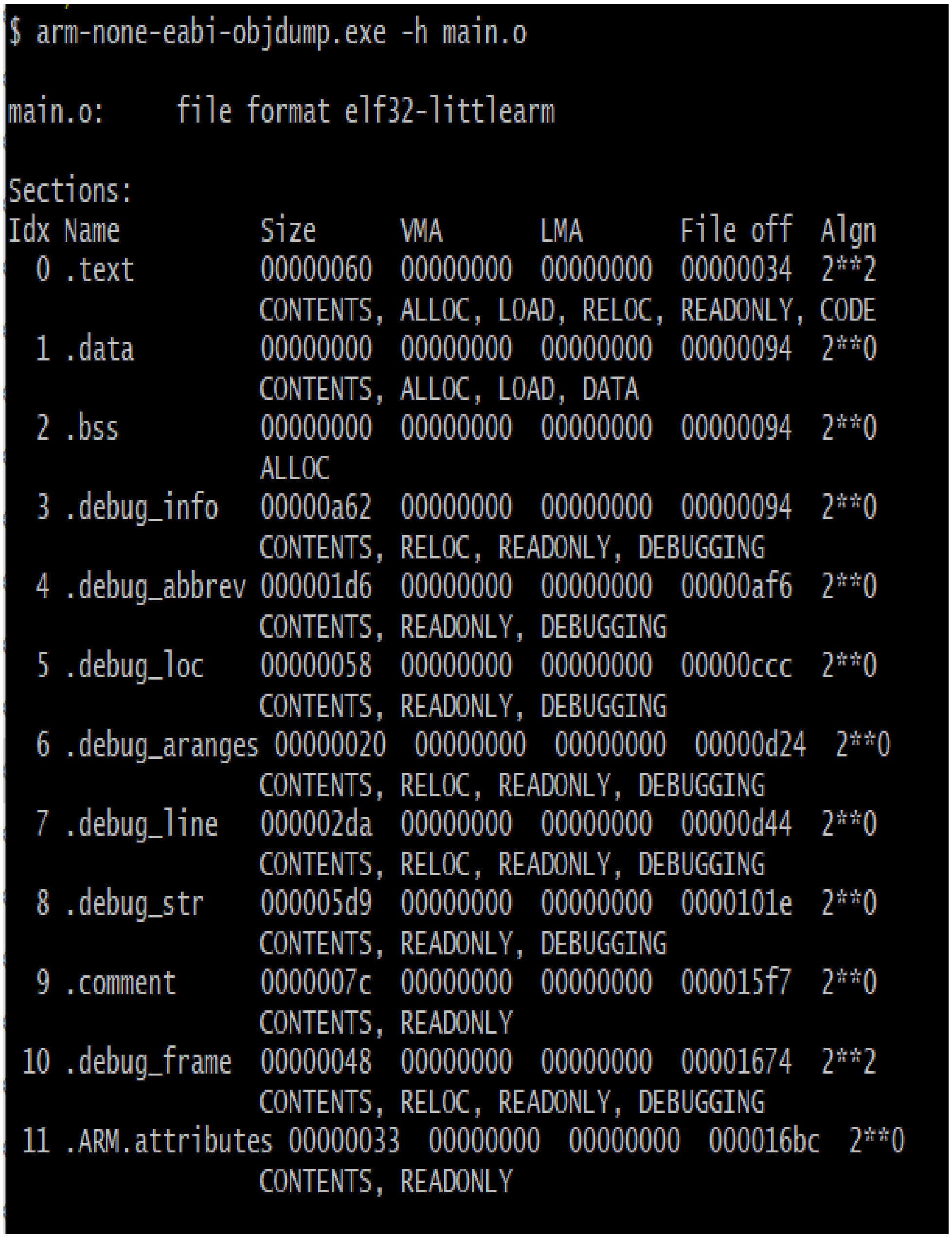
```
$ arm-none-eabi-objdump.exe -h pressure_sensor.o

pressure_sensor.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000038  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000006c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  0000006c  2**2
                  ALLOC
  3 .debug_info   000009f4  00000000  00000000  0000006c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001d2  00000000  00000000  00000a60  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000002c  00000000  00000000  00000c32  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000c5e  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000001f8  00000000  00000000  00000c7e  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000588  00000000  00000000  00000e76  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  000013fe  2**0
                  CONTENTS, READONLY
 10 .debug_frame  0000002c  00000000  00000000  0000147c  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  000014a8  2**0
                  CONTENTS, READONLY
```

- **Alarm.o**



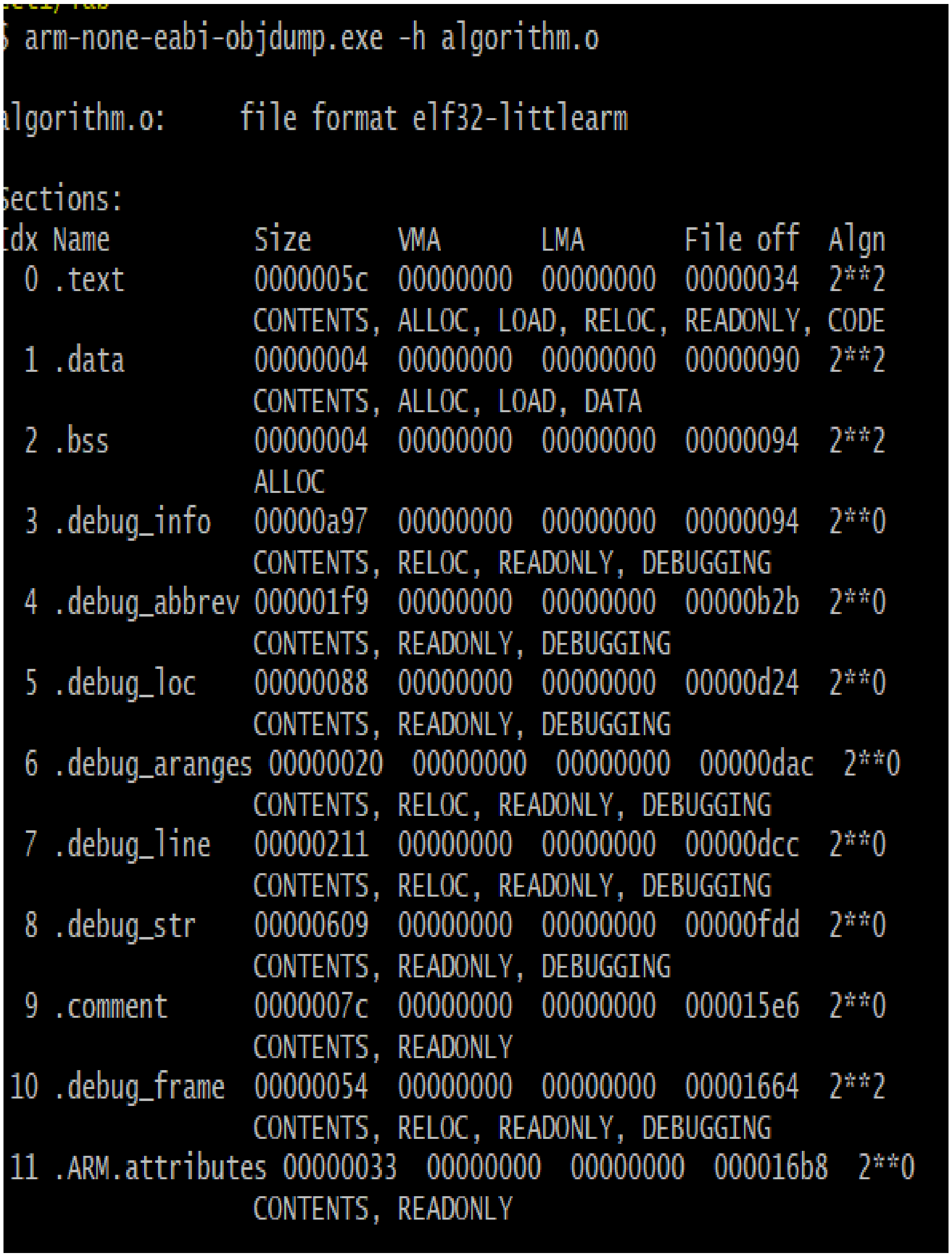
```
$ arm-none-eabi-objdump.exe -h Alarm.o

Alarm.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000006c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000a0  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000a0  2**0
                  ALLOC
  3 .debug_info   00000a07  00000000  00000000  000000a0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001e1  00000000  00000000  00000aa7  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000009c  00000000  00000000  00000c88  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000d24  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000001ee  00000000  00000000  00000d44  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000593  00000000  00000000  00000f32  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  000014c5  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000068  00000000  00000000  00001544  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  000015ac  2**0
                  CONTENTS, READONLY
```

- **Main.o**



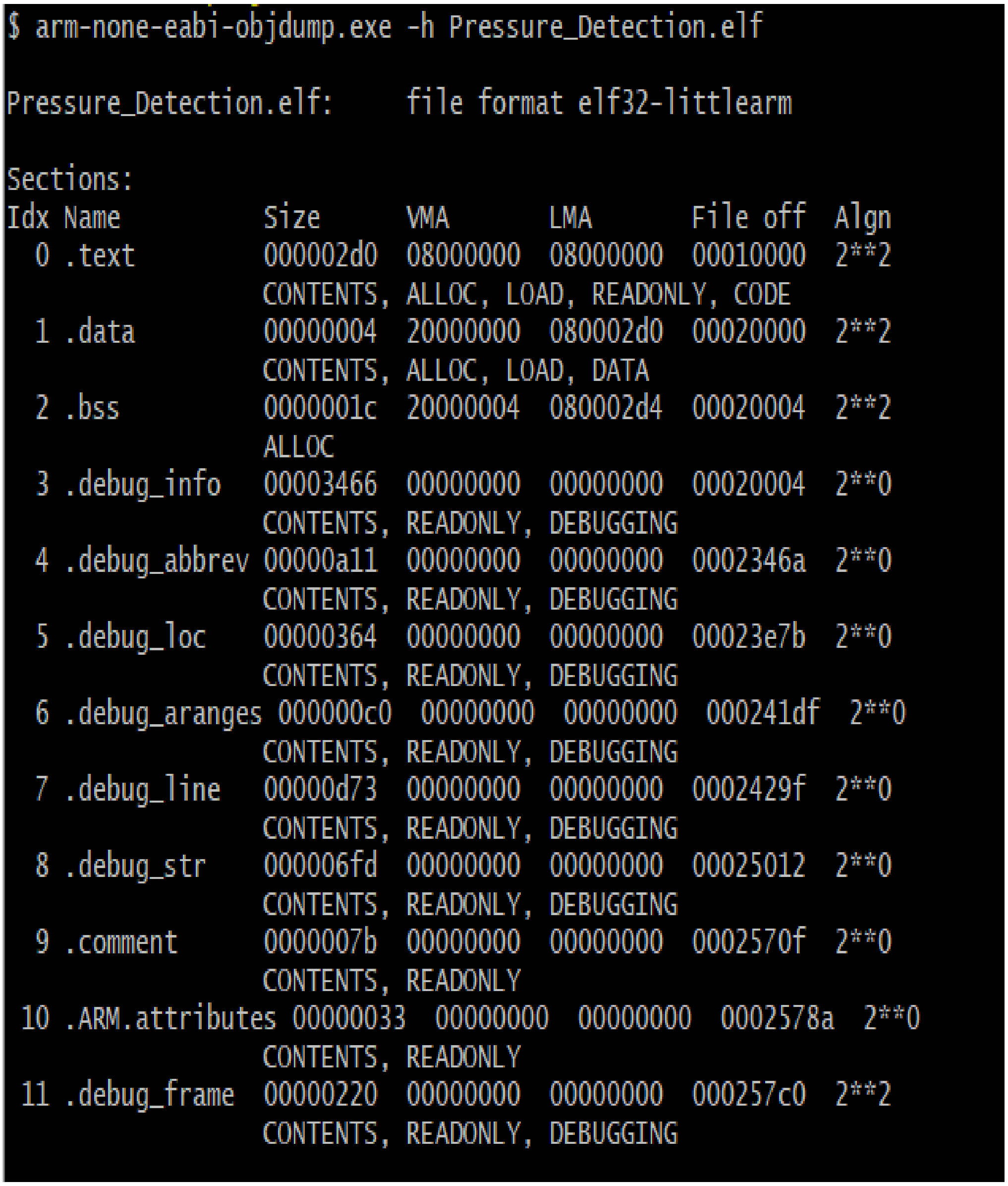
```
$ arm-none-eabi-objdump.exe -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000060  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000094  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000094  2**0
                  ALLOC
  3 .debug_info   00000a62  00000000  00000000  00000094  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001d6  00000000  00000000  00000af6  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000058  00000000  00000000  00000ccc  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000d24  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002da  00000000  00000000  00000d44  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000005d9  00000000  00000000  0000101e  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  000015f7  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000048  00000000  00000000  00001674  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  000016bc  2**0
                  CONTENTS, READONLY
```

- **Algorithm.o**



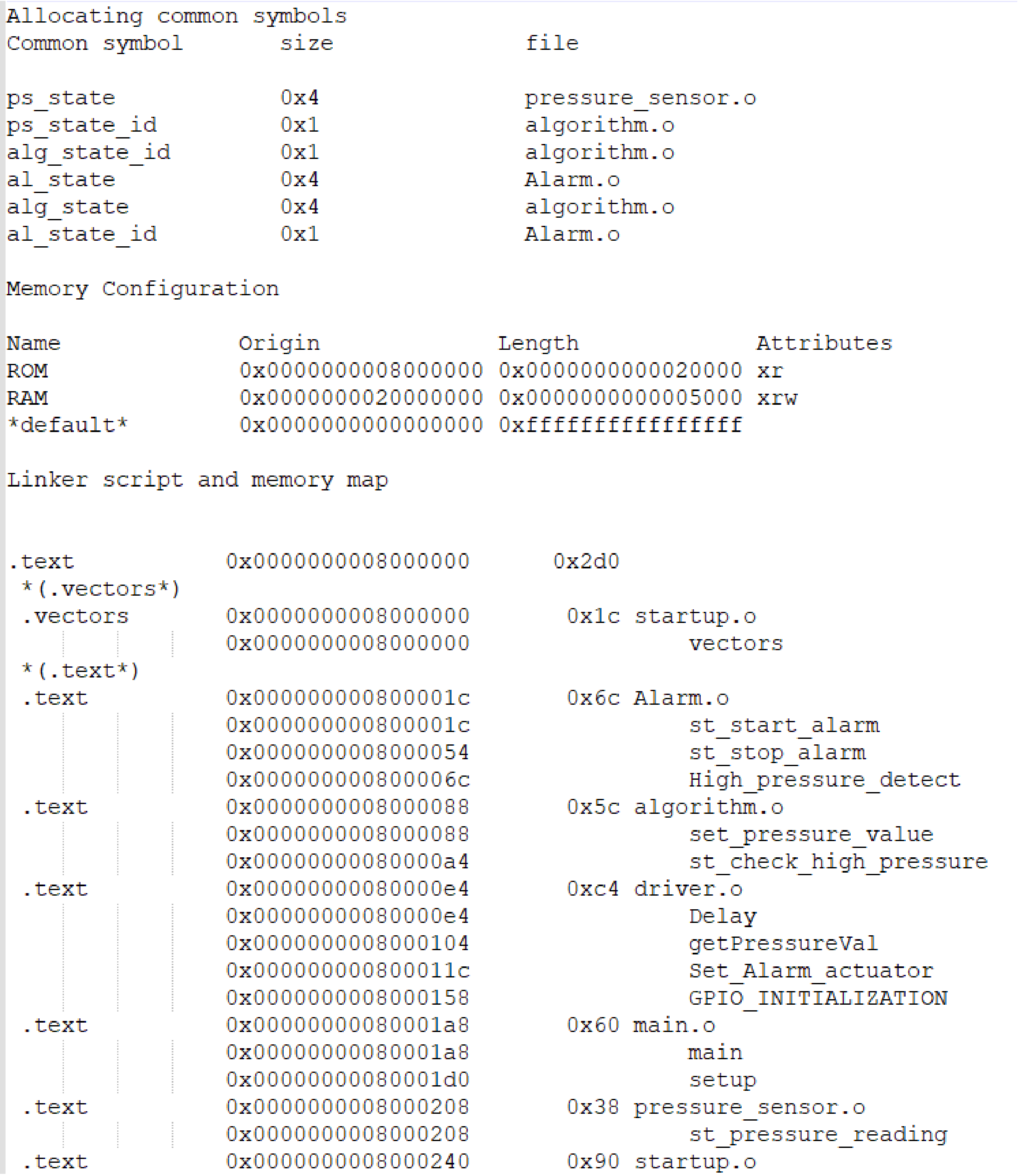
```
$ arm-none-eabi-objdump.exe -h algorithm.o

algorithm.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000005c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000004  00000000  00000000  00000090  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  00000094  2**2
                  ALLOC
  3 .debug_info   00000a97  00000000  00000000  00000094  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001f9  00000000  00000000  00000b2b  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000088  00000000  00000000  00000d24  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000dac  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000211  00000000  00000000  00000dcc  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000609  00000000  00000000  00000fdd  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  000015e6  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000054  00000000  00000000  00001664  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  000016b8  2**0
                  CONTENTS, READONLY
```

- **Pressure_Detection.elf**

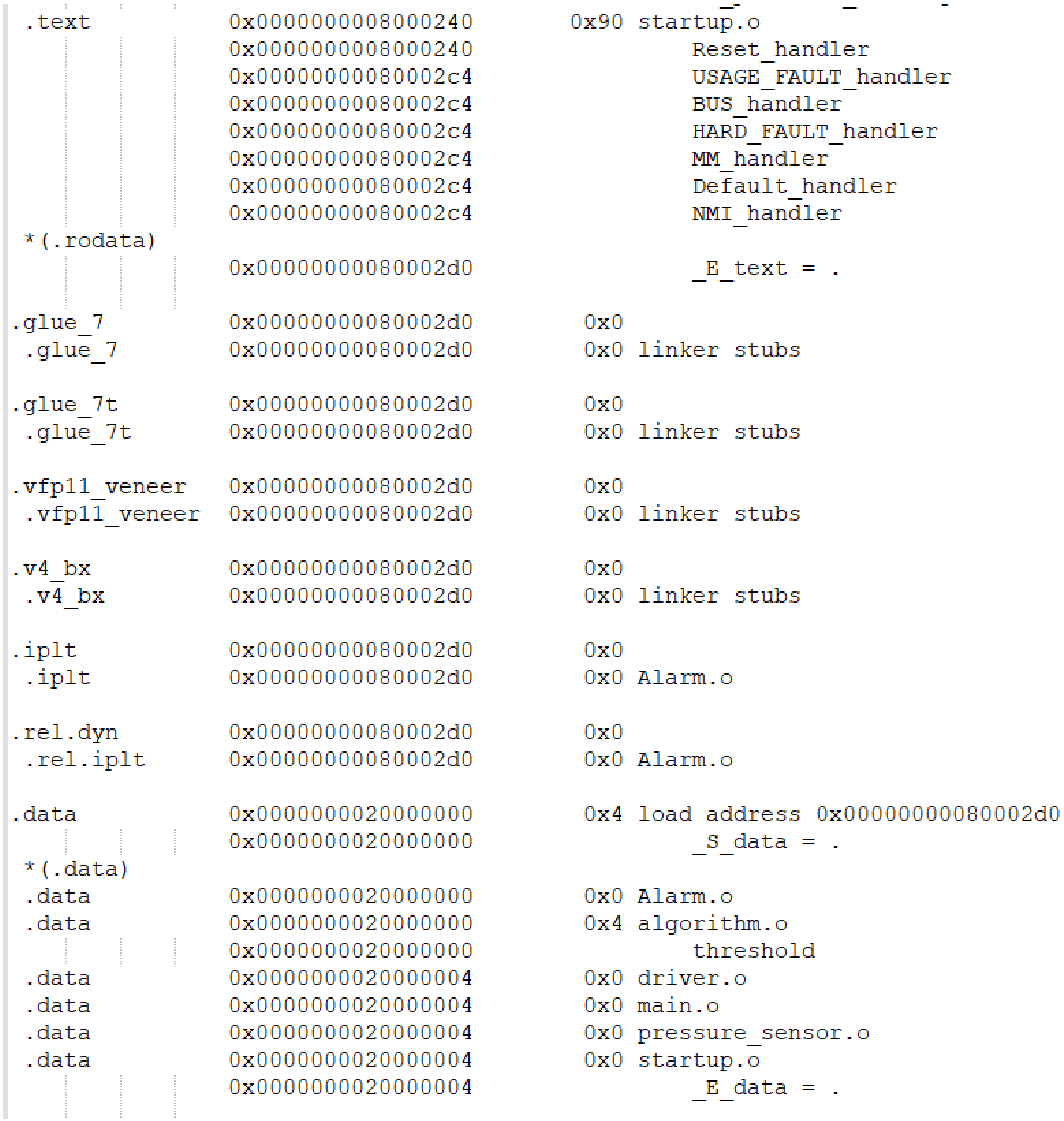


```
$ arm-none-eabi-objdump.exe -h Pressure_Detection.elf

Pressure_Detection.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000002d0  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  20000000  080002d0  00020000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          0000001c  20000004  080002d4  00020004  2**2
                  ALLOC
  3 .debug_info   00003466  00000000  00000000  00020004  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 00000a11  00000000  00000000  0002346a  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000364  00000000  00000000  00023e7b  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000c0  00000000  00000000  000241df  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000d73  00000000  00000000  0002429f  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    000006fd  00000000  00000000  00025012  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007b  00000000  00000000  0002570f  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  0002578a  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000220  00000000  00000000  000257c0  2**2
                  CONTENTS, READONLY, DEBUGGING
```

❖MapFile



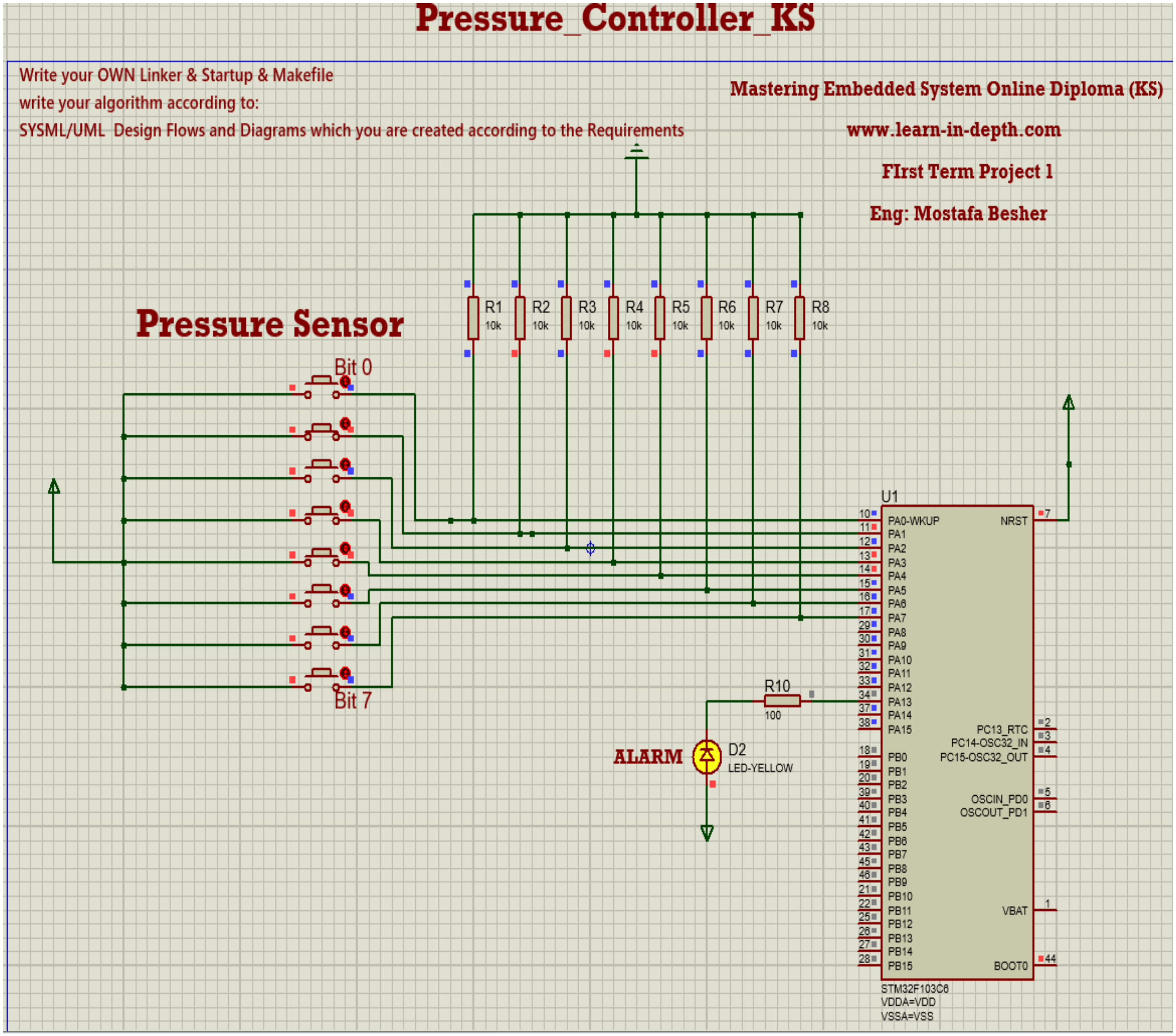
```
Allocating common symbols
Common symbol       size              file

ps_state            0x4               pressure_sensor.o
ps_state_id         0x1               algorithm.o
alg_state_id        0x1               algorithm.o
al_state            0x4               Alarm.o
alg_state           0x4               algorithm.o
al_state_id         0x1               Alarm.o

Memory Configuration

Name                Origin             Length             Attributes
ROM                 0x0000000008000000 0x0000000000020000 xr
RAM                 0x0000000020000000 0x0000000000005000 xrw
*default*           0x0000000000000000 0xffffffffffffffff

Linker script and memory map


.text               0x0000000008000000      0x2d0
 *(.vectors*)
 .vectors           0x0000000008000000       0x1c startup.o
                    0x0000000008000000                vectors
 *(.text*)
 .text              0x000000000800001c       0x6c Alarm.o
                    0x000000000800001c                st_start_alarm
                    0x0000000008000054                st_stop_alarm
                    0x000000000800006c                High_pressure_detect
 .text              0x0000000008000088       0x5c algorithm.o
                    0x0000000008000088                set_pressure_value
                    0x00000000080000a4                st_check_high_pressure
 .text              0x00000000080000e4       0xc4 driver.o
                    0x00000000080000e4                Delay
                    0x0000000008000104                getPressureVal
                    0x000000000800011c                Set_Alarm_actuator
                    0x0000000008000158                GPIO_INITIALIZATION
 .text              0x00000000080001a8       0x60 main.o
                    0x00000000080001a8                main
                    0x00000000080001d0                setup
 .text              0x0000000008000208       0x38 pressure_sensor.o
                    0x0000000008000208                st_pressure_reading
 .text              0x0000000008000240       0x90 startup.o
```



```
 .text          0x0000000008000240       0x90 startup.o
                0x0000000008000240                Reset_handler
                0x00000000080002c4                USAGE_FAULT_handler
                0x00000000080002c4                BUS_handler
                0x00000000080002c4                HARD_FAULT_handler
                0x00000000080002c4                MM_handler
                0x00000000080002c4                Default_handler
                0x00000000080002c4                NMI_handler
 *(.rodata)
                0x00000000080002d0                _E_text = .

.glue_7         0x00000000080002d0        0x0
 .glue_7        0x00000000080002d0        0x0 linker stubs

.glue_7t        0x00000000080002d0        0x0
 .glue_7t       0x00000000080002d0        0x0 linker stubs

.vfp11_veneer   0x00000000080002d0        0x0
 .vfp11_veneer  0x00000000080002d0        0x0 linker stubs

.v4_bx          0x00000000080002d0        0x0
 .v4_bx         0x00000000080002d0        0x0 linker stubs

.iplt           0x00000000080002d0        0x0
 .iplt          0x00000000080002d0        0x0 Alarm.o

.rel.dyn        0x00000000080002d0        0x0
 .rel.iplt      0x00000000080002d0        0x0 Alarm.o

.data           0x0000000020000000        0x4 load address 0x00000000080002d0
                0x0000000020000000                _S_data = .
 *(.data)
 .data          0x0000000020000000        0x0 Alarm.o
 .data          0x0000000020000000        0x4 algorithm.o
                0x0000000020000000                threshold
 .data          0x0000000020000004        0x0 driver.o
 .data          0x0000000020000004        0x0 main.o
 .data          0x0000000020000004        0x0 pressure_sensor.o
 .data          0x0000000020000004        0x0 startup.o
                0x0000000020000004                _E_data = .
```

❖Proteus Simulation

- **When pressure > 20 bars , alarm started**
  - **Pressure in simulation equals 26 bars**

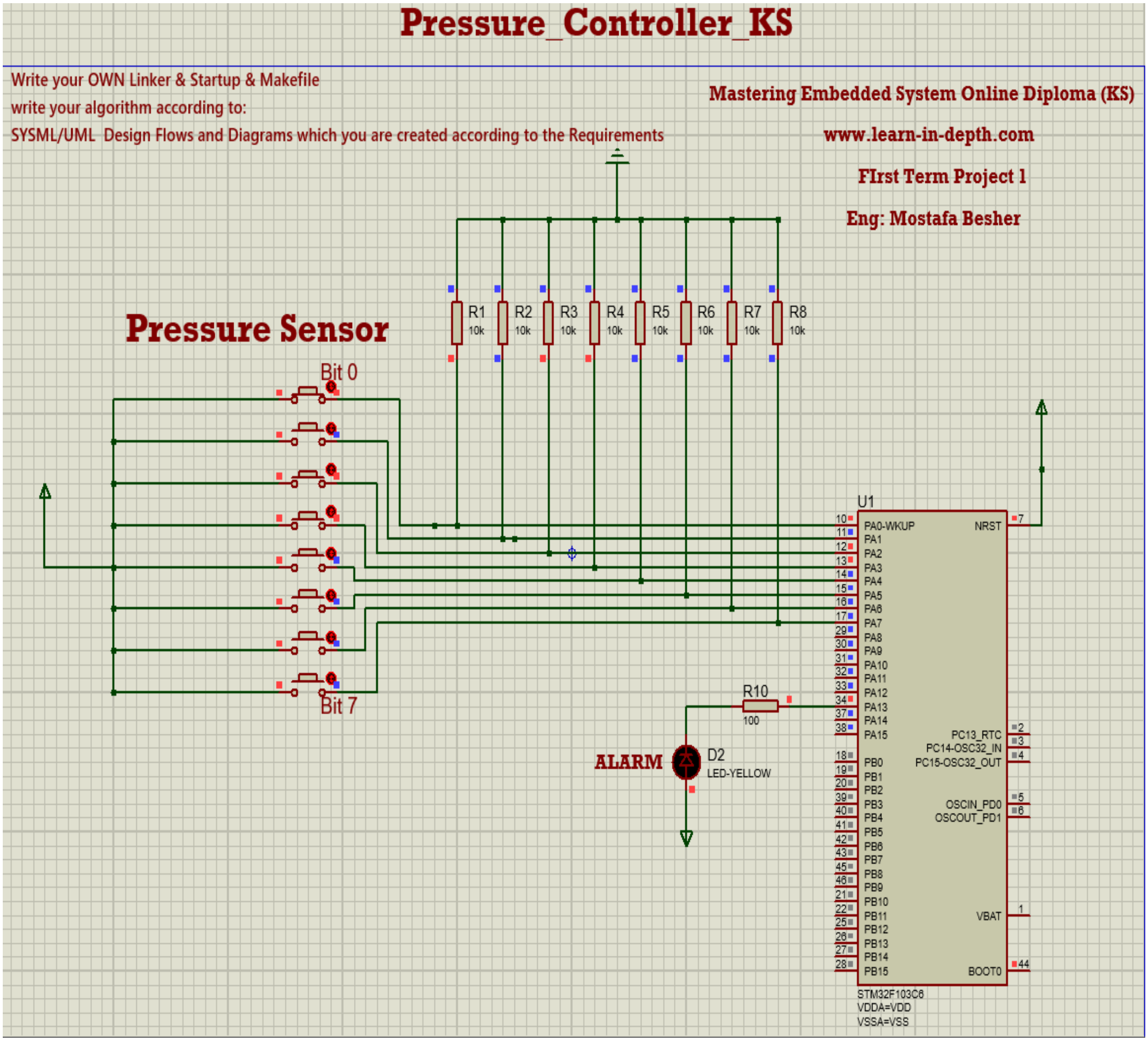


Pressure_Controller_KS

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
Eng: Mostafa Besher

- **When pressure <= 20 bars, alarm stopped**
    - **Pressure in simulation equals 13 bars**



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