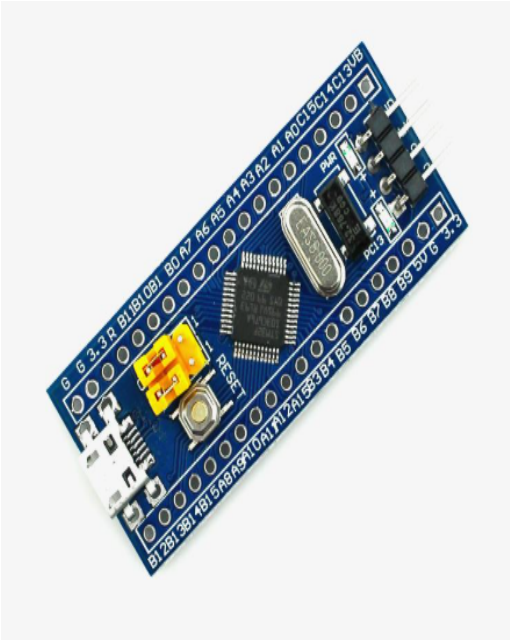


Pressure Detection System

1 NOV 2025

Eng - Belal Hani Abu Sabha

Learn in Depth Company

Profile:

https://www.learn-in-depth-store.com/account/applystar

Github:

https://github.com/belalsabha/Master_Embedded_Systems

Linkedin:

https://www.linkedin.com/in/belal-sabha-06ba33349/

Content——————————————————————————1

OverView and Case Study——————————————————2

Method————————————————————————————3

Requirements——————————————————————————4

Space Exploration————————————————————————5

System Analysis——————————————————————————6

System Design————————————————————————————9

Implementation C Code——————————————————————13

Symbols & Sections——————————————————————22

Mabfile.map——————————————————————————26

Simulation————————————————————————————28

Conclusion————————————————————————————30

Overview

The Pressure Controller *System* is an embedded control system designed to monitor pressure in Room . The system continuously reads data from a pressure sensor detect if the value is more or less than threshold to control in alarm status .

The project demonstrates a complete embedded system design cycle followed by the implementation of modular C code and hardware-level integration on the STM32F103C6 microcontroller.

Case Study :

1-Specifications

1. A pressure controller informs the crew of a cabin when the pressure exceeds 20 bars in the cabin
2. The alarm duration equals 60 seconds.

2-Assumptions :

1-The controller set up and shutdown procedures are not modeled

2-The controller maintenance is not modeled

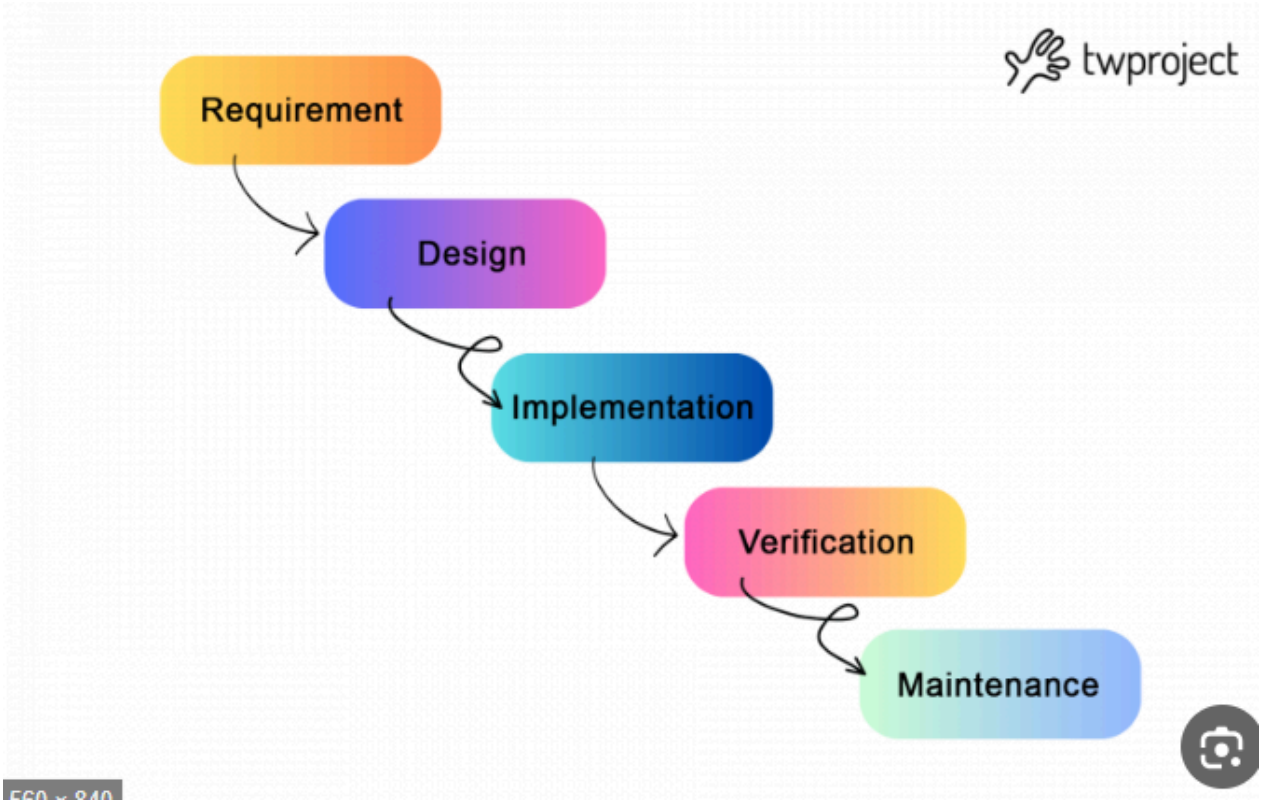
3-The pressure sensor never fails

4-The alarm never fails

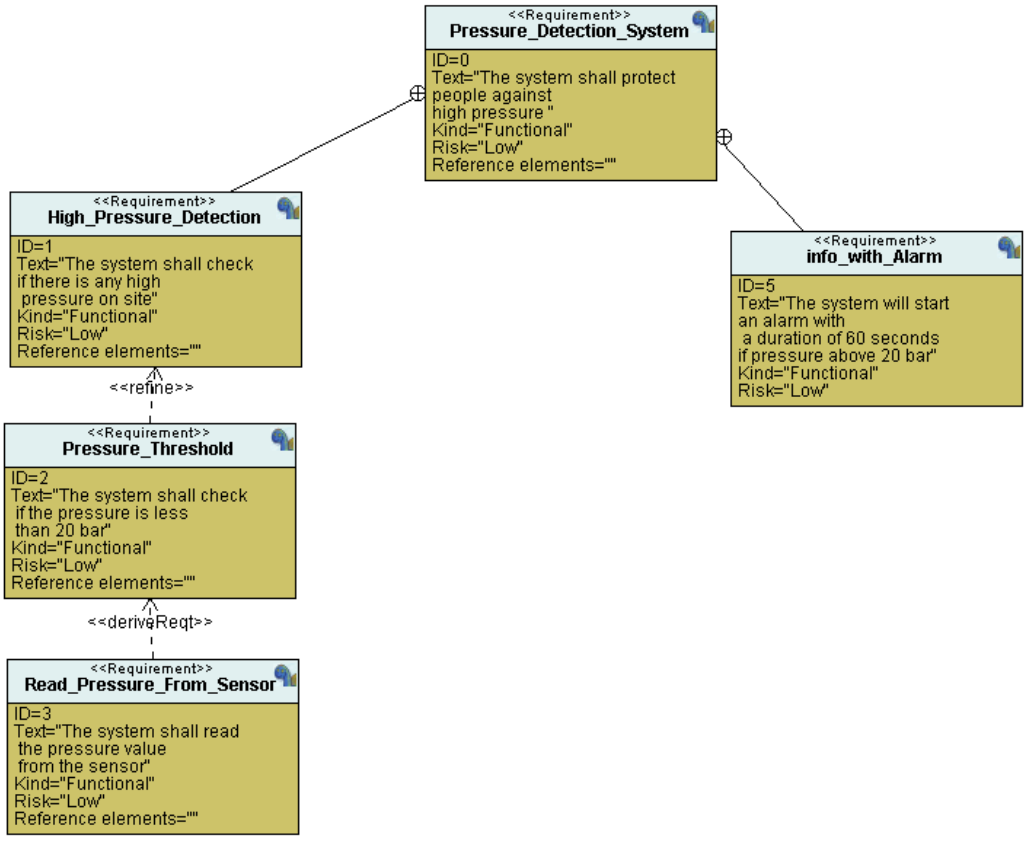
5-The controller never faces power cut

Method :

I. **we will be working on the waterfall method**

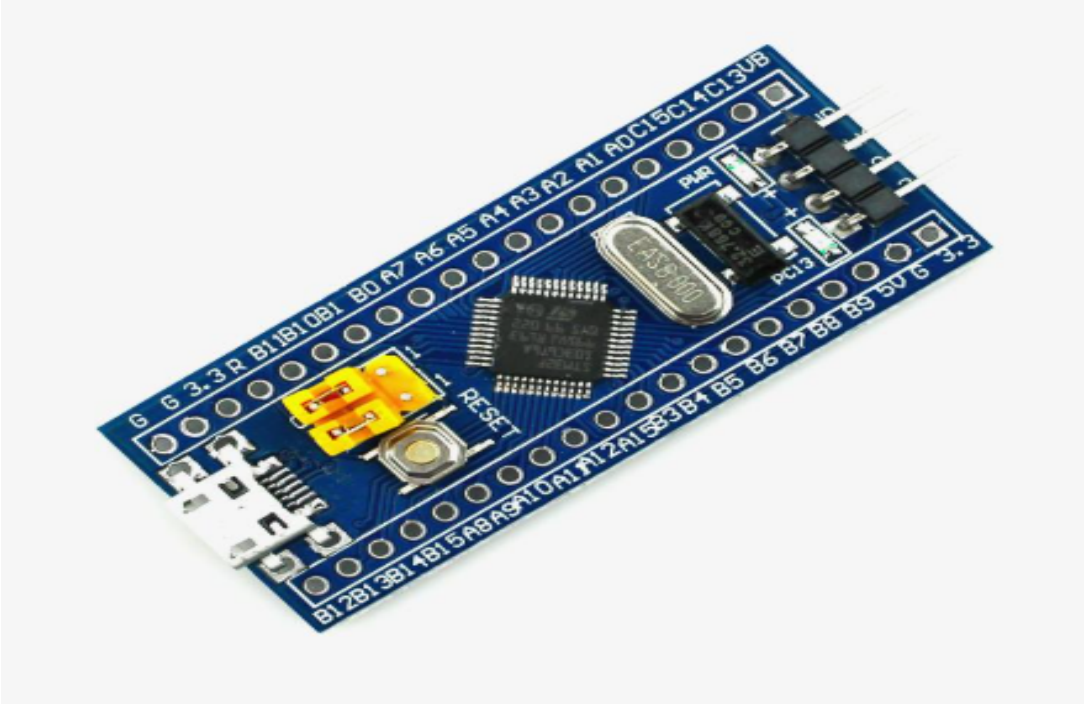


Requirements :



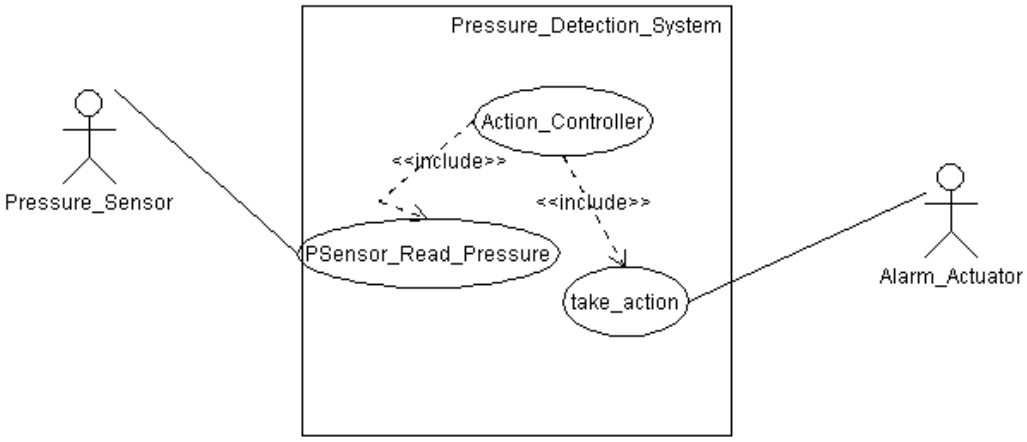
Space Exploration :

1-Use STM32F103C6 Microcontroller with Cortex-M3

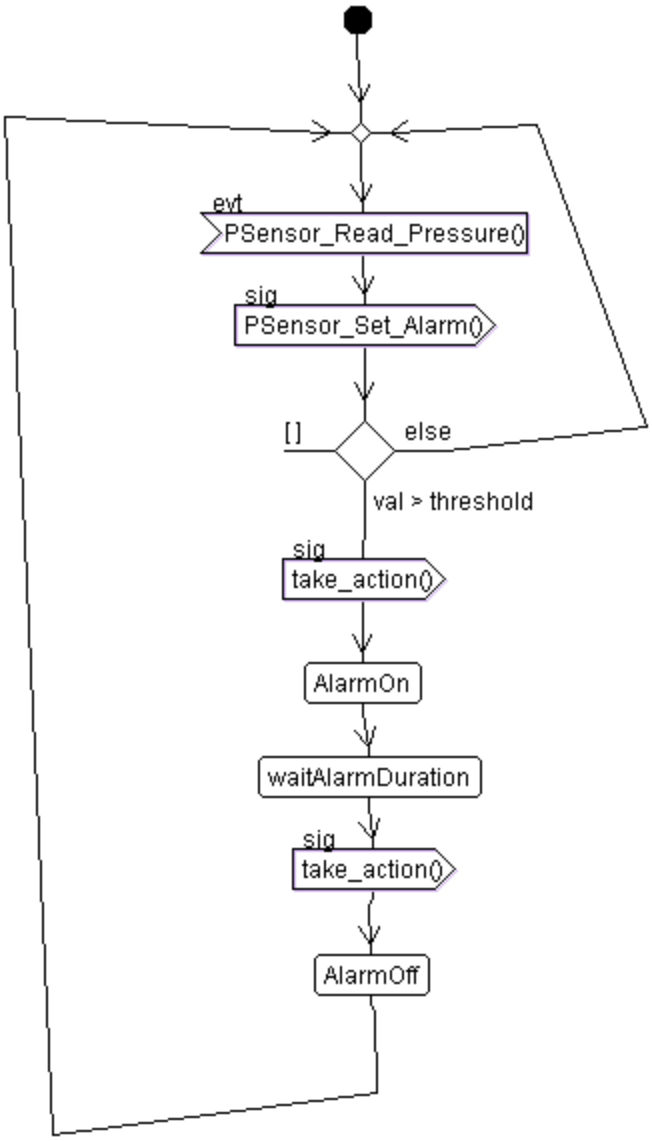


System Analysis :

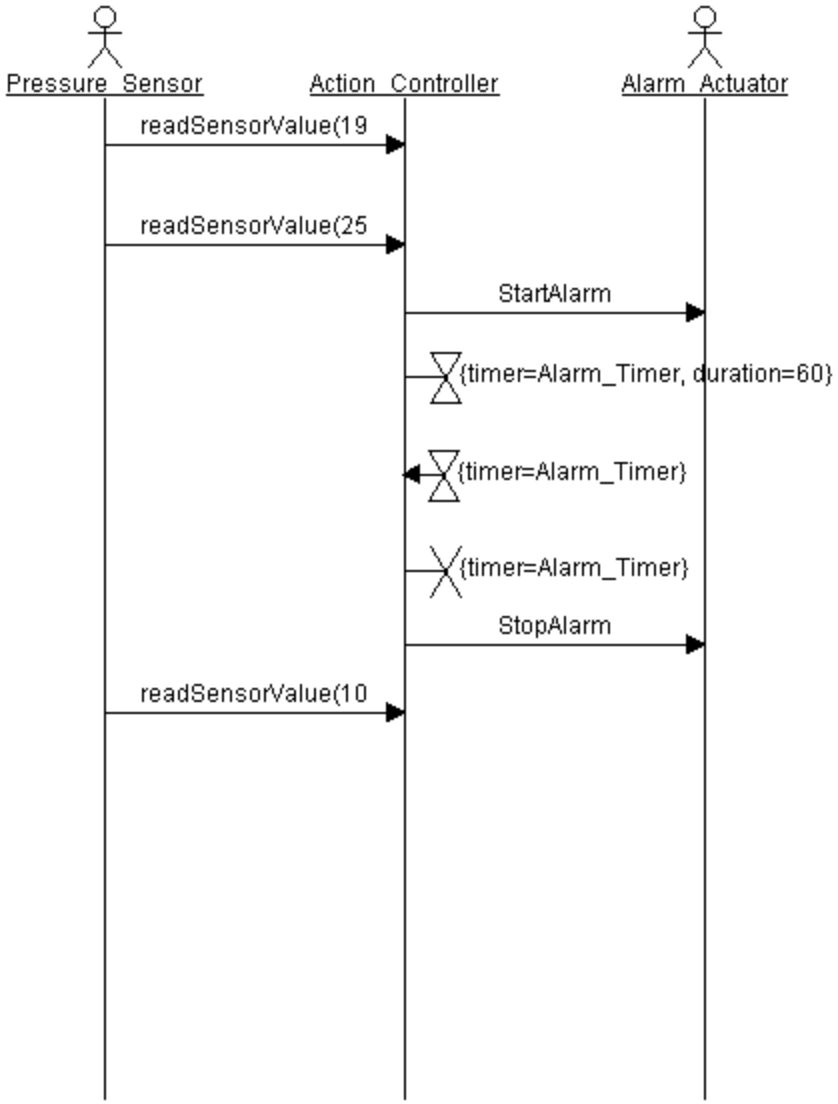
1-Use-Case Diagram :



2-Activity Diagram :

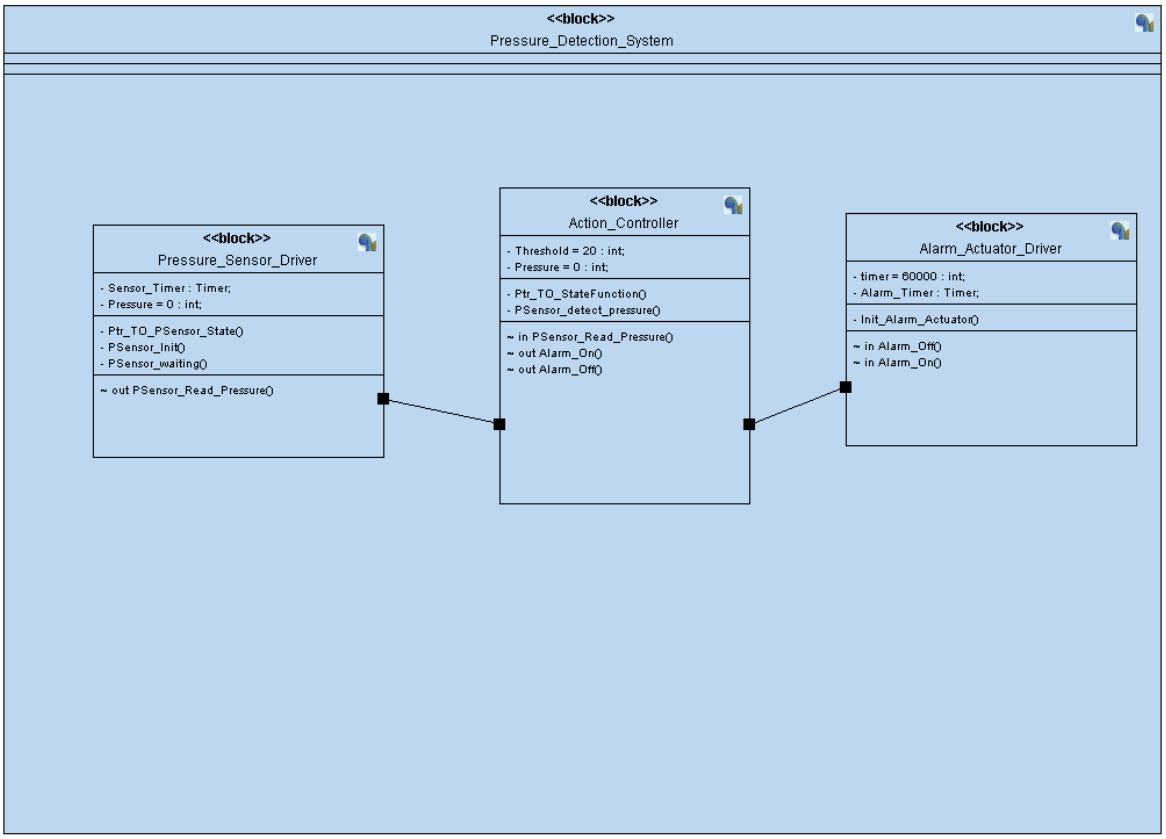


3-Sequence Diagram :



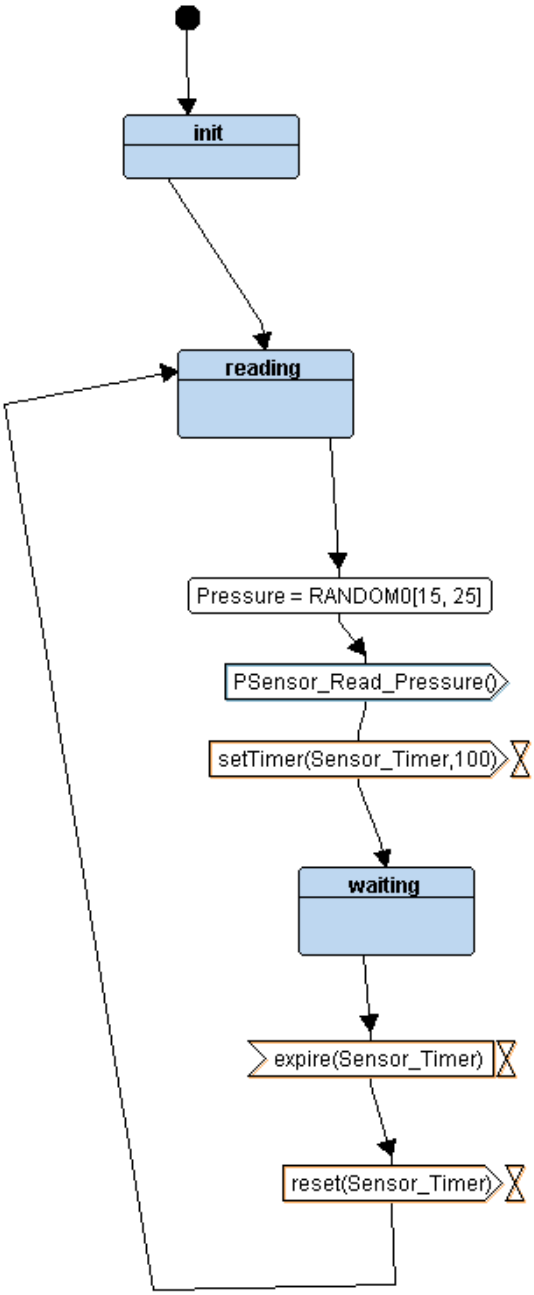
System Design :

1-Block Diagram :

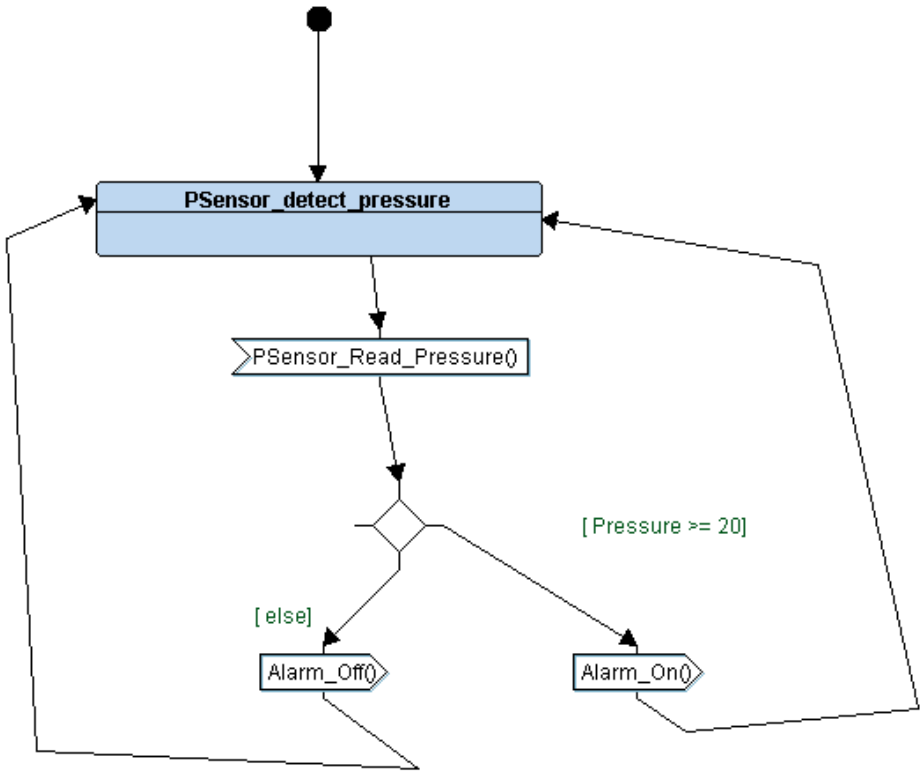


2-State Machine for every Module :

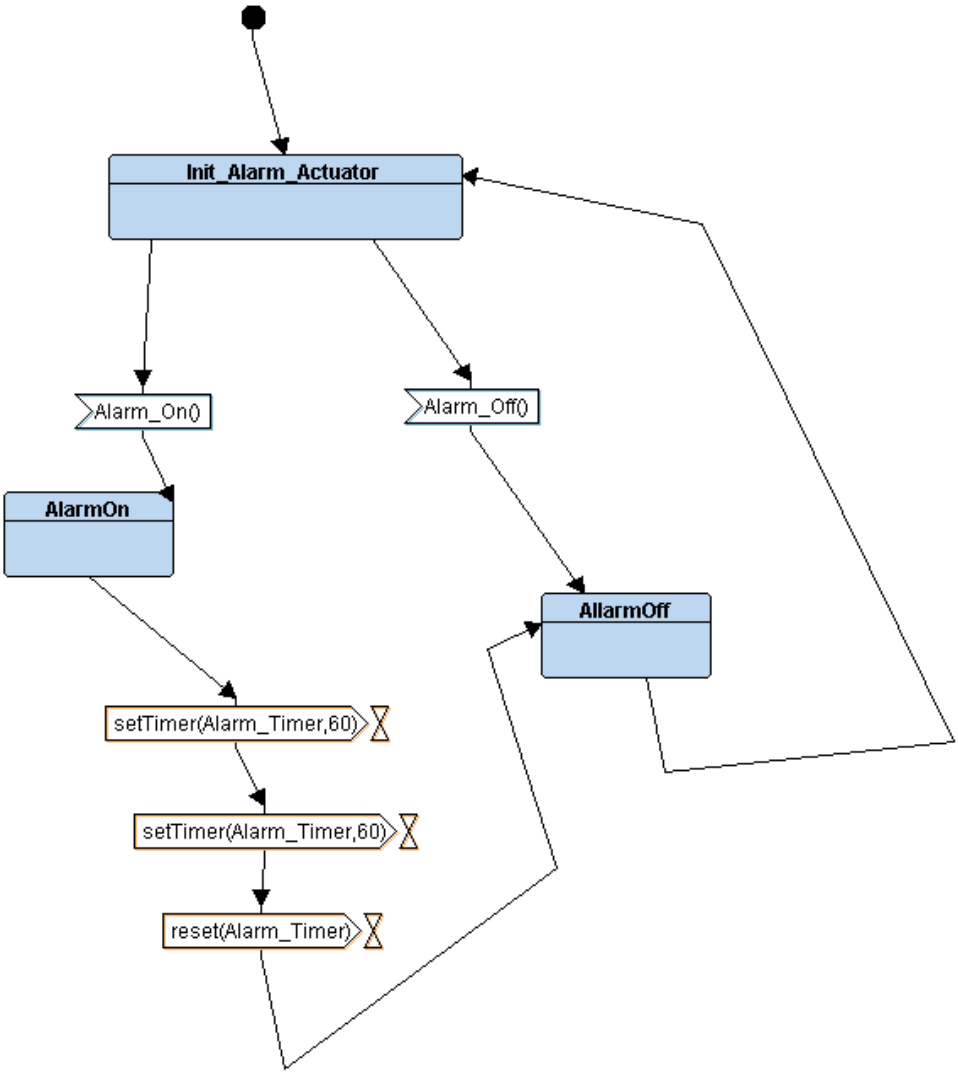
A-Pressure_Sensor_Driver :



B-Action_Controller :



C-Alarm_Actuator_Driver :



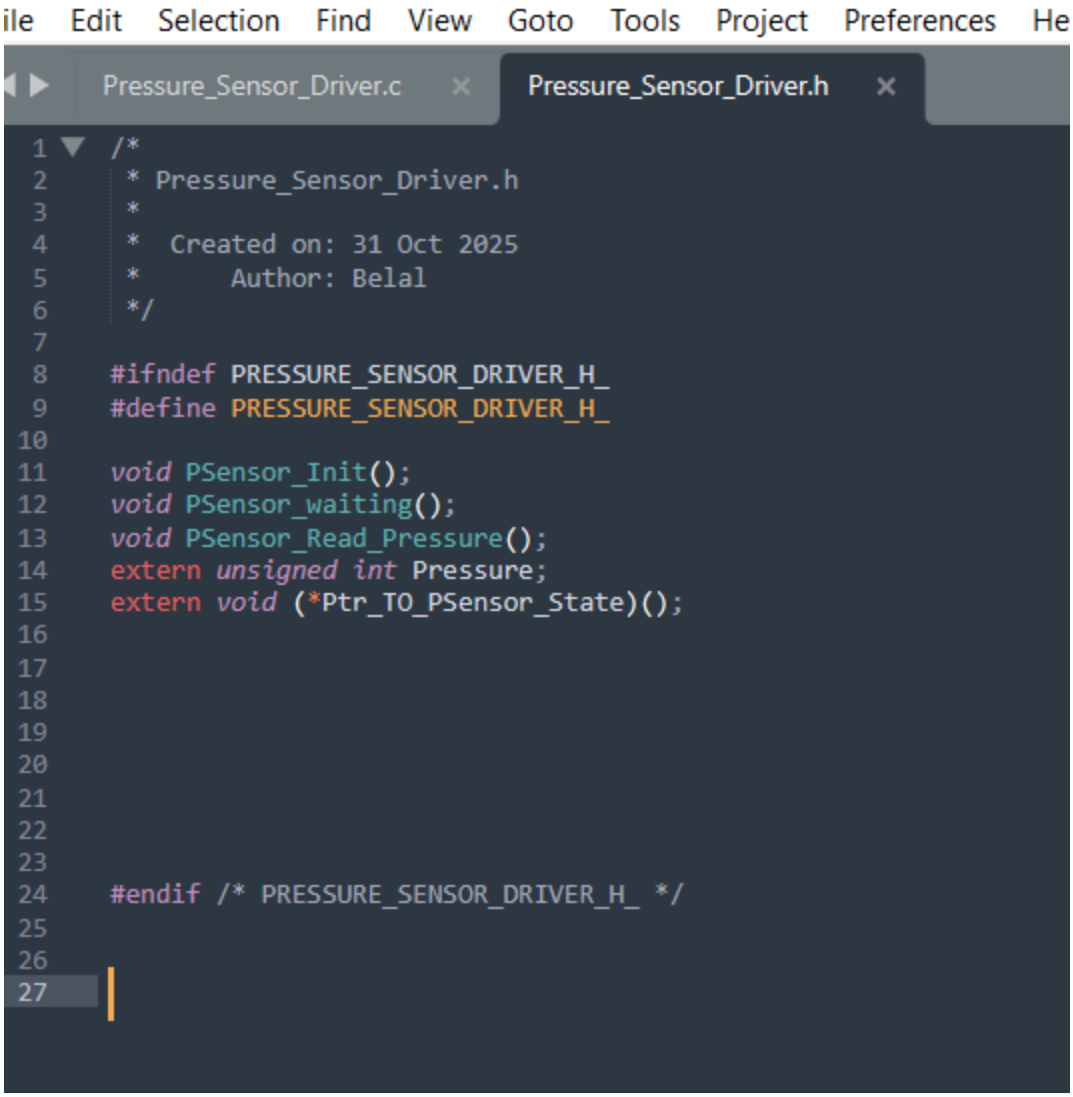
Implementation C Code :

Pressure_Sensor_Driver.c

Pressure_Sensor_Driver.c

```c
/*
 * Pressure_Sensor_Driver.c
 *
 *  Created on: 31 Oct 2025
 *      Author: Belal
 */



#include "Pressure_Sensor_Driver.h"
#include "driver.h"
unsigned int Pressure= 0 ;
void (*Ptr_TO_PSensor_State)();

void PSensor_Init(){
    Ptr_TO_PSensor_State=PSensor_Read_Pressure;
}
void PSensor_Read_Pressure(){
    Pressure= (unsigned int)getPressureVal();
    PSensor_waiting();
    Ptr_TO_PSensor_State=PSensor_Read_Pressure;

}

void PSensor_waiting(){
    int delay=1000;
    Delay(delay);
}
```

Pressure_Sensor_Driver.h



Alarm_Actuator_Driver.c

Pressure_Sensor_Driver.c | Pressure_Sensor_Driver.h | Alarm_Actuator_Driver.c

```c
/*
 * Alarm_Actuator_Driver.c
 *
 *  Created on: 31 Oct 2025
 *      Author: Belal
 */
#include "Alarm_Actuator_Driver.h"
#include "driver.h"
unsigned int time=600000;

void Alarm_Off(){
    Set_Alarm_actuator(1);
}
void Alarm_On(){
    Set_Alarm_actuator(0);
    Delay(time);
    Alarm_Off();

}
void Init_Alarm_Actuator(){

}
```

Alarm_Actuator_Driver.h

Alarm_Actuator_Driver.h

```c
/*
 * Alarm_Actuator_Driver.h
 *
 *  Created on: 31 Oct 2025
 *      Author: Belal
 */

#ifndef ALARM_ACTUATOR_DRIVER_H_
#define ALARM_ACTUATOR_DRIVER_H_

#include "driver.h"
void Alarm_Off();
void Alarm_On();
void Init_Alarm_Actuator();
extern void (*Ptr_TO_Alarm_State);


#endif /* ALARM_ACTUATOR_DRIVER_H_ */
```

Action_Controller.c

```c
/*
 * Action_Controller.c
 *
 *  Created on: 31 Oct 2025
 *      Author: Belal
 */

#include "Action_Controller.h"
#include "Pressure_Sensor_Driver.h"
#include "Alarm_Actuator_Driver.h"


static const unsigned int Threshold = 20;

void (*Ptr_TO_StateFunction)();


void PSensor_detect_pressure(){

    if(Pressure < Threshold )
        Alarm_Off();
    else
        Alarm_On();

}
```

Action_Controller.h

```c
 * Action_Controller.h
 *
 *  Created on: 31 Oct 2025
 *      Author: Belal
 */

#ifndef ACTION_CONTROLLER_H_
#define ACTION_CONTROLLER_H_
#include "Pressure_Sensor_Driver.h"
#include "Alarm_Actuator_Driver.h"
extern void (*Ptr_TO_StateFunction)();
void set_alarm();
void reset_alarm();
void PSensor_detect_pressure();


#endif /* ACTION_CONTROLLER_H_ */
```

Main.c

```c
/*
 * main.c
 *
 *  Created on: 31 Oct 2025
 *      Author: Belal
 */


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

#include "driver.h"
#include "Pressure_Sensor_Driver.h"
#include "Action_Controller.h"
#include "Alarm_Actuator_Driver.h"


void setup(){
    Ptr_TO_StateFunction=PSensor_detect_pressure;
    Ptr_TO_PSensor_State=PSensor_Read_Pressure;
    GPIO_INITIALIZATION();

}

int main (){
    setup();
    while (1)
    {
        Ptr_TO_PSensor_State();
        Ptr_TO_StateFunction();



    }

}

```

Driver.c

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;
}

```

Driver.h

driver.c driver.h

```
#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 ();
```

LinkerScript.ld

LinkerScript.ld

```
/*
* linker.ld
*
*  Created on: 31 Oct 2025
*      Author: Belal
*/

MEMORY {
    flash(rx) : ORIGIN = 0x08000000 , LENGTH = 128k
    sram(rwx) : ORIGIN = 0x20000000 , LENGTH = 20k
}

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

    .data : {
        Start_Data = . ;
        *(.data)
        End_Data = . ;
    }>sram AT>flash

    .bss : {
         Start_Bss = . ;
        *(.bss)
         . = ALIGN(4) ;
         End_Bss = . ;
         . = ALIGN(4) ;
         . = . + 0x1000;  ;
         Stack_Top = . ;
    }>sram
}
```

Startup.c

LinkerScript.ld | startup.c

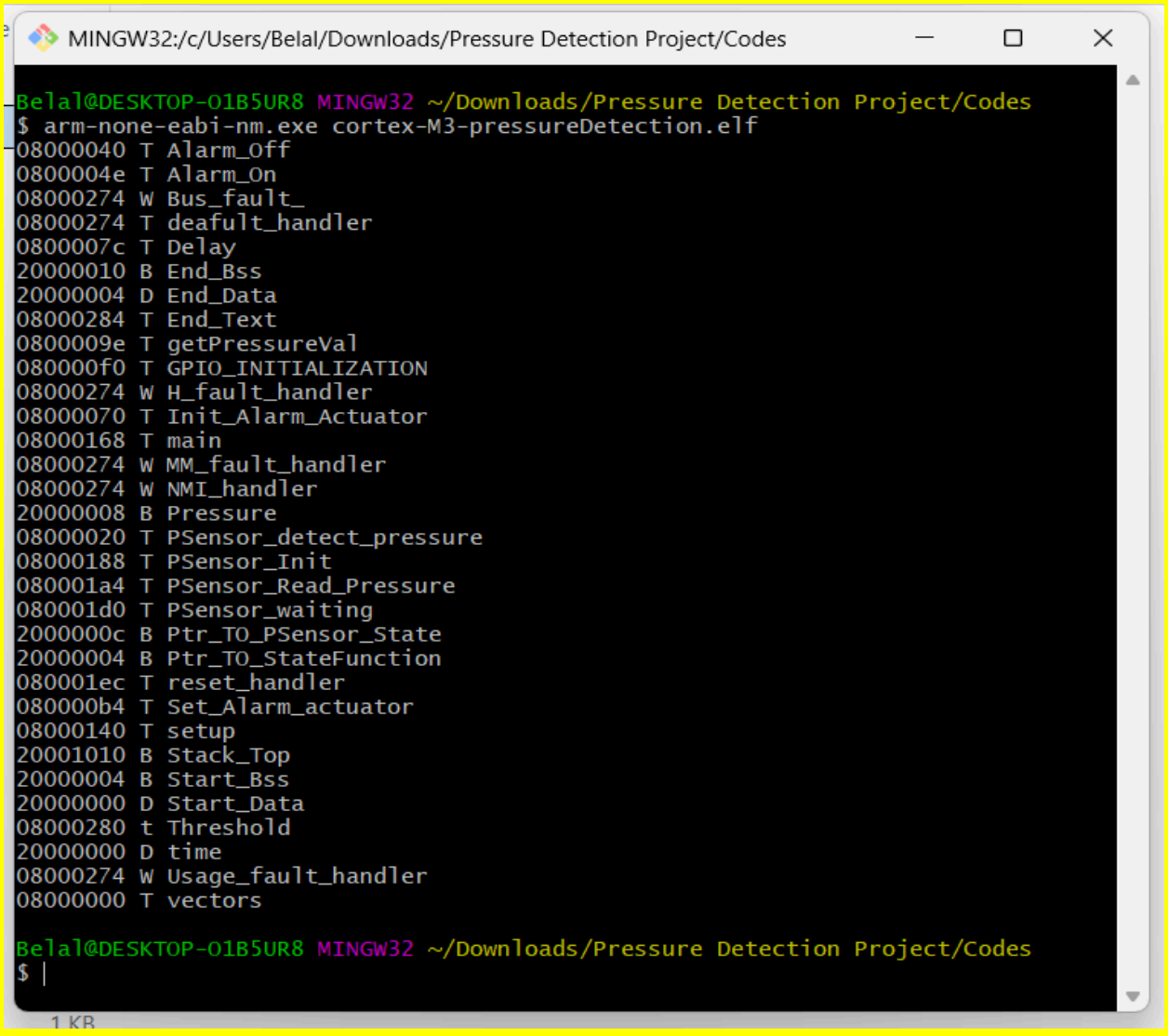
```c
/*
 * @file      startup.c
 * Created on: 31 Oct 2025
 *  Author: Belal
 */
typedef unsigned int uint32_t ;
typedef unsigned char uint8_t;
#define Start_Stack_SP 0x20001000
extern int main(void);
void reset_handler(void);
void deafult_handler(void);
void NMI_handler(void)                    __attribute__ ((weak, alias("deafult_handler")));;
void H_fault_handler(void)                __attribute__ ((weak, alias("deafult_handler")));;
void MM_fault_handler(void)               __attribute__ ((weak, alias("deafult_handler")));;
void Bus_fault_(void)                     __attribute__ ((weak, alias("deafult_handler")));;
void Usage_fault_handler(void)            __attribute__ ((weak, alias("deafult_handler")));;
uint32_t vectors[]  __attribute__ ((section(".vectors"))) ={
Start_Stack_SP,
(uint32_t)&reset_handler,
(uint32_t)&deafult_handler,
(uint32_t)&NMI_handler,
(uint32_t)&H_fault_handler,
(uint32_t)&MM_fault_handler,
(uint32_t)&Bus_fault_,
(uint32_t)&Usage_fault_handler,
};
extern uint32_t End_Text ;
extern uint32_t Start_Data;
extern uint32_t End_Data;
extern uint32_t Start_Bss;
extern uint32_t End_Bss;
extern uint32_t Stack_Top;
void reset_handler(void){
    // copying data from flash to sram
    uint32_t Data_Size  = (uint8_t*)&End_Data - (uint8_t*)&Start_Data;
    uint8_t *Src = (uint8_t*)&End_Text;
    uint8_t *Des = (uint8_t*)&Start_Data;
    for (int i = 0; i < Data_Size; ++i)
    {
        *((uint8_t*)Des++) = *((uint8_t*)Src++);
    // create .bss section and init with zero
        uint32_t Bss_Size  = (uint8_t*)&End_Bss - (uint8_t*)&Start_Bss;
         Des = (uint8_t*)&Start_Bss;
    for (int i = 0; i < Bss_Size; ++i)
    {
        *Des++ = 0;
    }
// go to main in main.c file
main();
}
}
void deafult_handler(void){
reset_handler();
}
```

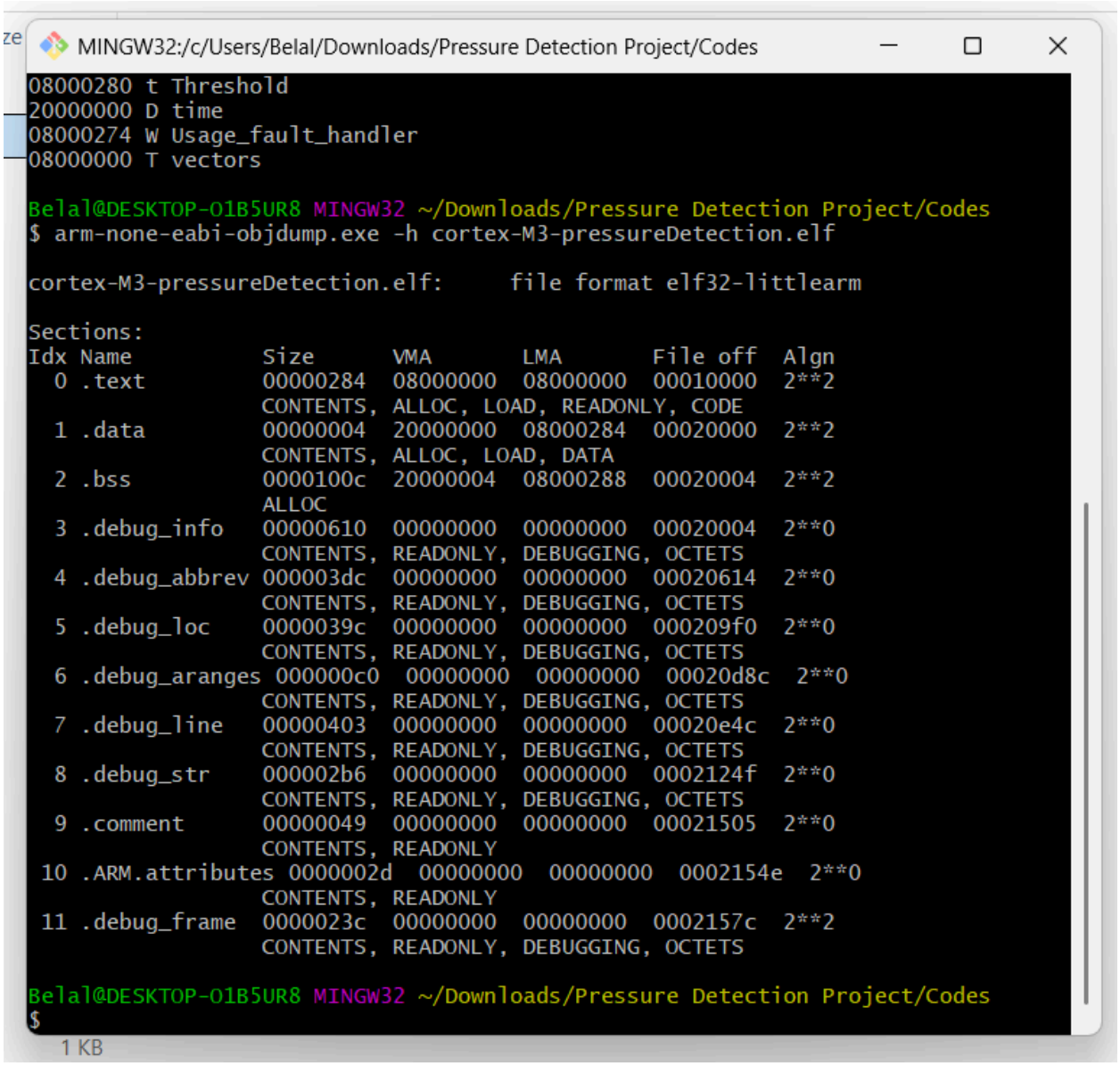
Makefile

```
#@copyright : Belal Hani Sabha

CC=arm-none-eabi-

CFLAGS=-gdwarf-2 -mcpu=cortex-m3

INCS=-I .

LIBS=

SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)

AS=$(wildcard *.s)
ASO=$(AS:.s=.o)




PROJECT_NAME=cortex-M3-pressureDetection

all: $(PROJECT_NAME).bin
    @echo "build is done"

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

$(PROJECT_NAME).elf: $(OBJ) $(ASO)
    $(CC)ld.exe -T LinkerScript.ld $(LIBS) $(OBJ) $(ASO) -o $@ -Map=mab_file.map

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

clean_all:
    rm  *.o *.elf *.bin *.map
    @echo "everything clean..."

clean:
    rm  *.bin *.elf
    @echo "clean .elf and .bin files"
```

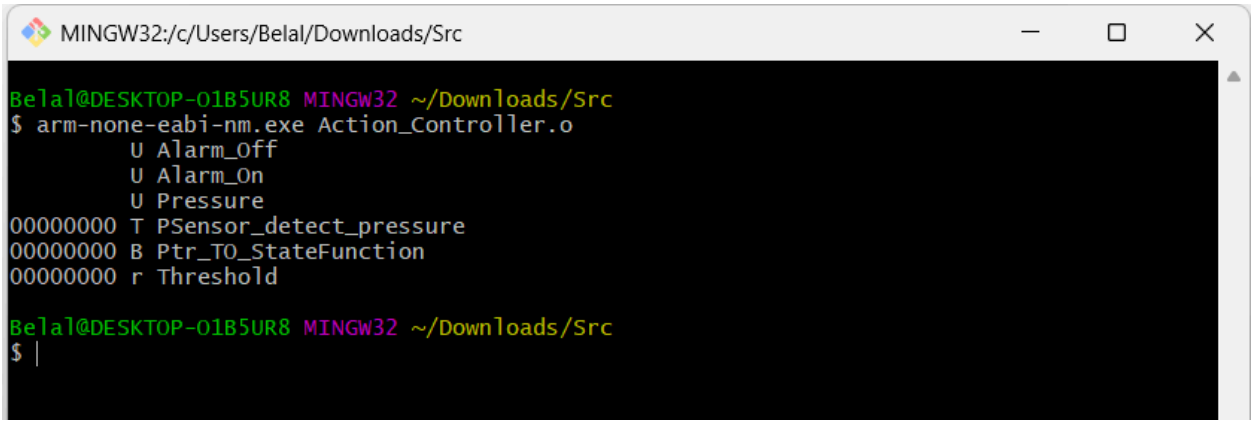
Symbols & Sections :

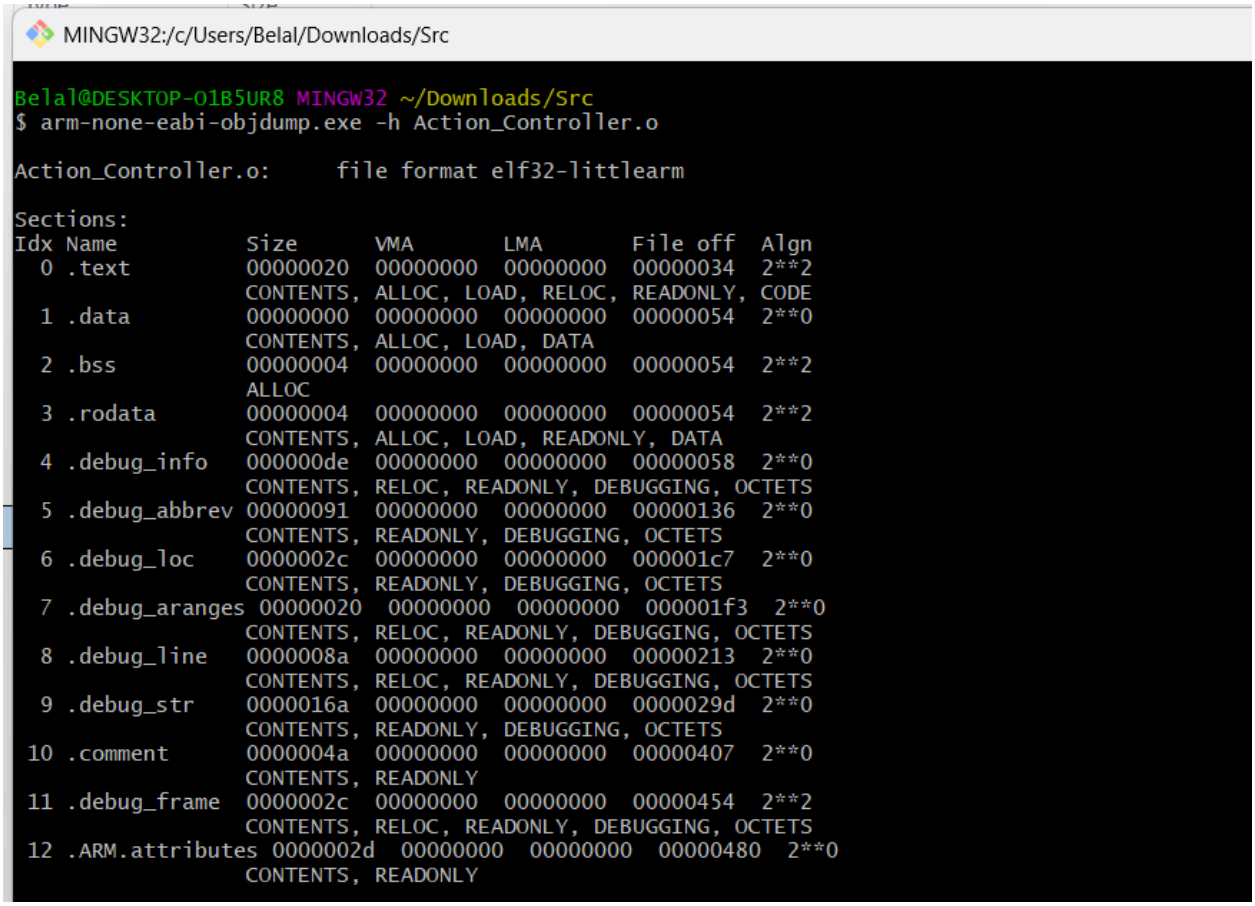
cortex-M3-pressureDetection.elf



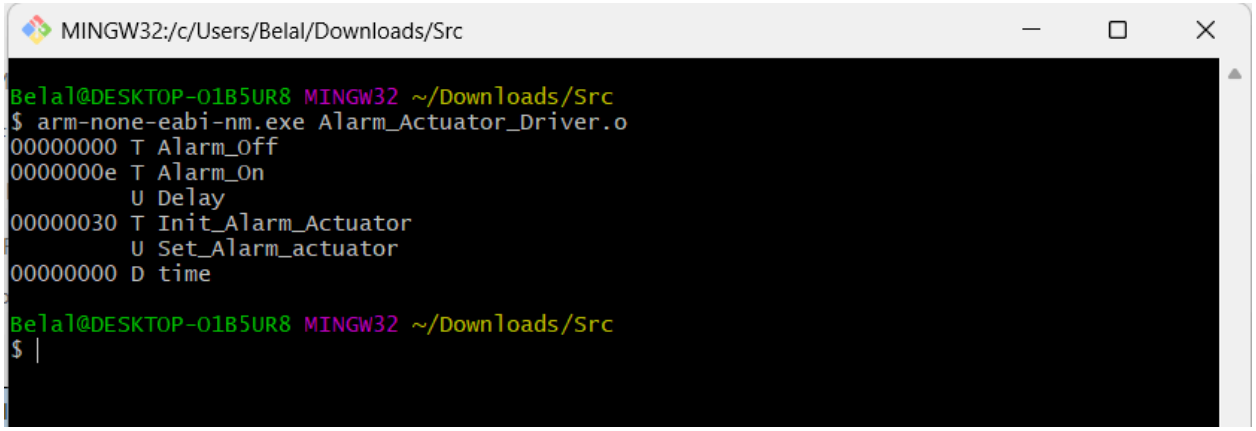

MINGW32:/c/Users/Belal/Downloads/Pressure Detection Project/Codes
08000280 t Threshold
20000000 D time
08000274 W Usage_fault_handler
08000000 T vectors
Belal@DESKTOP-O1B5UR8 MINGW32 ~/Downloads/Pressure Detection Project/Codes
$ arm-none-eabi-objdump.exe -h cortex-M3-pressureDetection.elf
cortex-M3-pressureDetection.elf: file format elf32-littlearm
Sections:
Idx Name Size VMA LMA File off Algn
0 .text 00000284 08000000 08000000 00010000 2**2
CONTENTS, ALLOC, LOAD, READONLY, CODE
1 .data 00000004 20000000 08000284 00020000 2**2
CONTENTS, ALLOC, LOAD, DATA
2 .bss 0000100c 20000004 08000288 00020004 2**2
ALLOC
3 .debug_info 00000610 00000000 00000000 00020004 2**0
CONTENTS, READONLY, DEBUGGING, OCTETS
4 .debug_abbrev 000003dc 00000000 00000000 00020614 2**0
CONTENTS, READONLY, DEBUGGING, OCTETS
5 .debug_loc 0000039c 00000000 00000000 000209f0 2**0
CONTENTS, READONLY, DEBUGGING, OCTETS
6 .debug_aranges 000000c0 00000000 00000000 00020d8c 2**0
CONTENTS, READONLY, DEBUGGING, OCTETS
7 .debug_line 00000403 00000000 00000000 00020e4c 2**0
CONTENTS, READONLY, DEBUGGING, OCTETS
8 .debug_str 000002b6 00000000 00000000 0002124f 2**0
CONTENTS, READONLY, DEBUGGING, OCTETS
9 .comment 00000049 00000000 00000000 00021505 2**0
CONTENTS, READONLY
10 .ARM.attributes 0000002d 00000000 00000000 0002154e 2**0
CONTENTS, READONLY
11 .debug_frame 0000023c 00000000 00000000 0002157c 2**2
CONTENTS, READONLY, DEBUGGING, OCTETS
Belal@DESKTOP-O1B5UR8 MINGW32 ~/Downloads/Pressure Detection Project/Codes
$
1 KB

Action_Controller.c



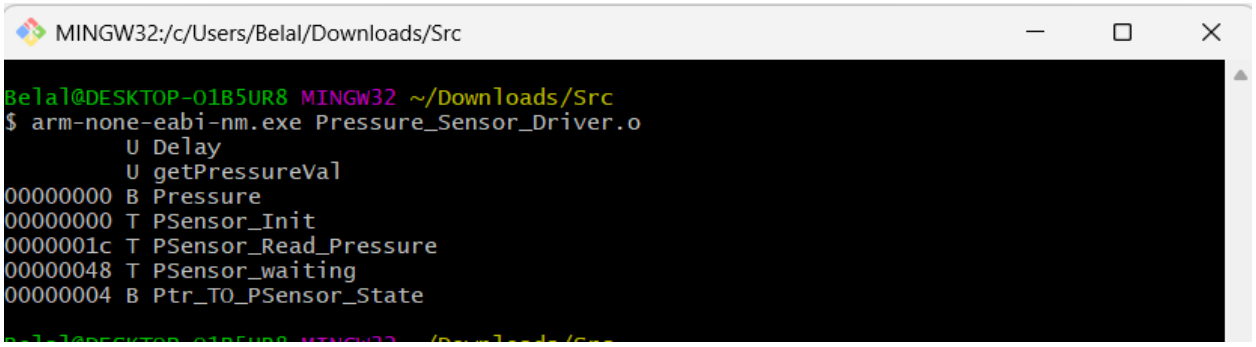


Alarm_Actuator_Driver.c



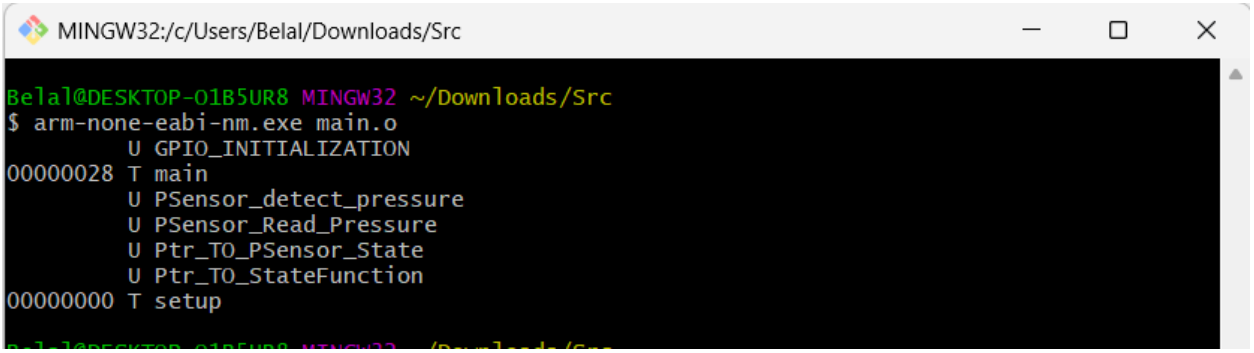

```
MINGW32:/c/Users/Belal/Downloads/Src

Belal@DESKTOP-O1B5UR8 MINGW32 ~/Downloads/Src
$ arm-none-eabi-objdump.exe -h Alarm_Actuator_Driver.o

Alarm_Actuator_Driver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000003c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000004  00000000  00000000  00000070  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000074  2**0
                  ALLOC
  3 .debug_info   000000cf  00000000  00000000  00000074  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  4 .debug_abbrev 0000006d  00000000  00000000  00000143  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  5 .debug_loc    0000009c  00000000  00000000  000001b0  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  6 .debug_aranges 00000020  00000000  00000000  0000024c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  7 .debug_line   00000060  00000000  00000000  0000026c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  8 .debug_str    0000015a  00000000  00000000  000002cc  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  9 .comment      0000004a  00000000  00000000  00000426  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000068  00000000  00000000  00000470  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 11 .ARM.attributes 0000002d  00000000  00000000  000004d8  2**0
                  CONTENTS, READONLY
```

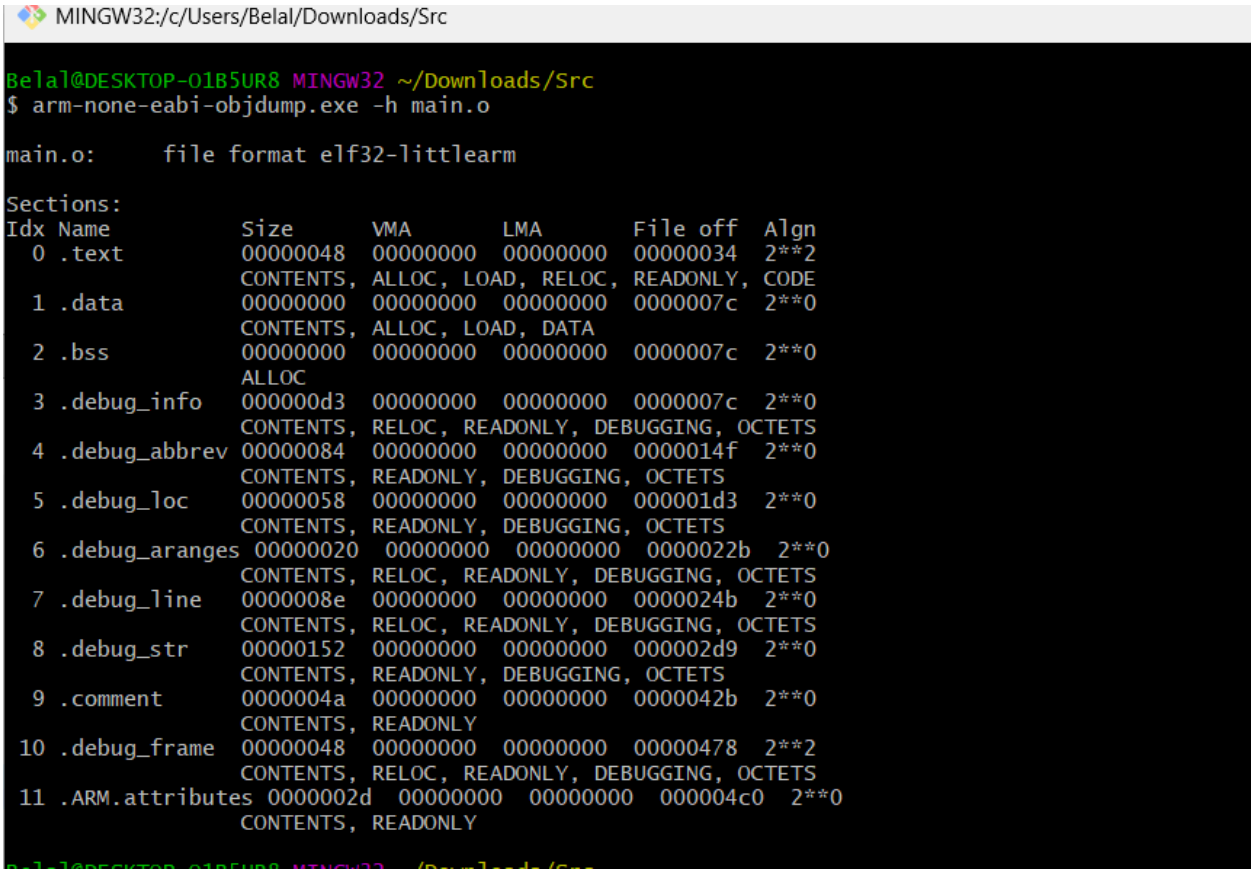
Pressure_Sensor_Driver.c



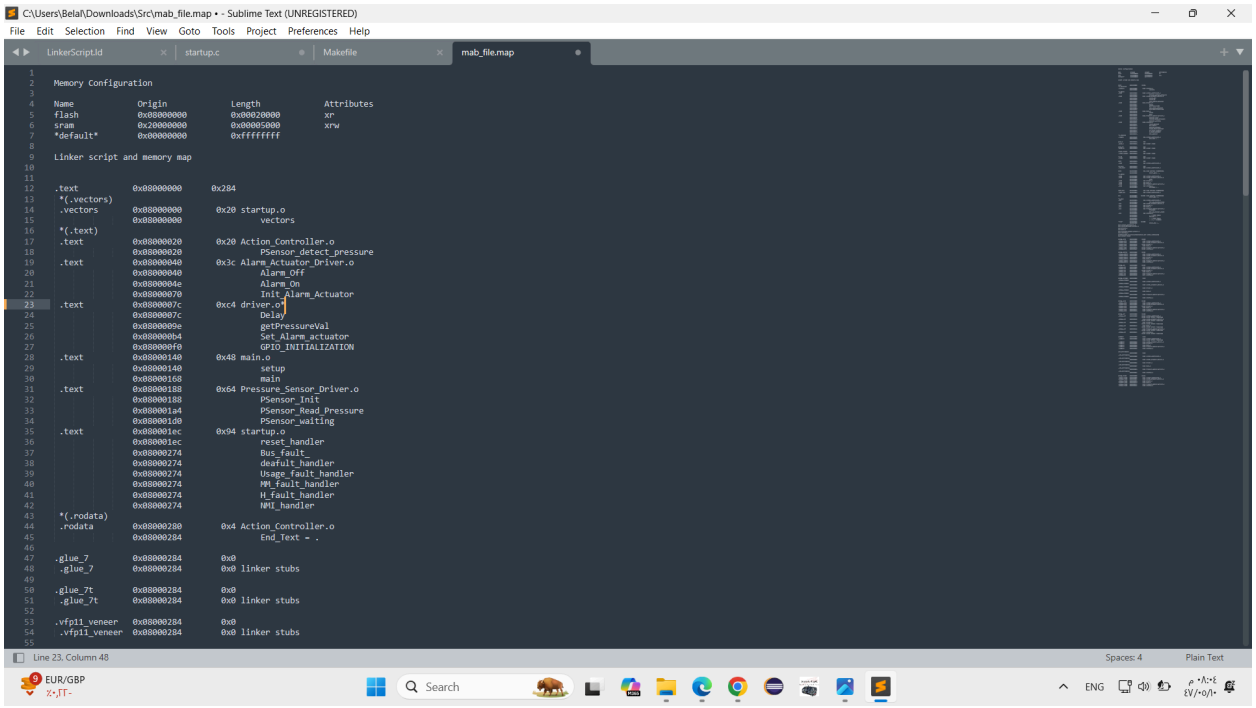

```
Belal@DESKTOP-O1B5UR8 MINGW32 ~/Downloads/Src
$ arm-none-eabi-objdump.exe -h Pressure_Sensor_Driver.o

Pressure_Sensor_Driver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000064  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000098  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000008  00000000  00000000  00000098  2**2
                  ALLOC
  3 .debug_info   00000115  00000000  00000000  00000098  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  4 .debug_abbrev 000000bc  00000000  00000000  000001ad  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  5 .debug_loc    000000c0  00000000  00000000  00000269  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  6 .debug_aranges 00000020  00000000  00000000  00000329  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  7 .debug_line   0000008d  00000000  00000000  00000349  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  8 .debug_str    00000186  00000000  00000000  000003d6  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  9 .comment      0000004a  00000000  00000000  0000055c  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000070  00000000  00000000  000005a8  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 11 .ARM.attributes 0000002d  00000000  00000000  00000618  2**0
                  CONTENTS, READONLY
```

Main.c

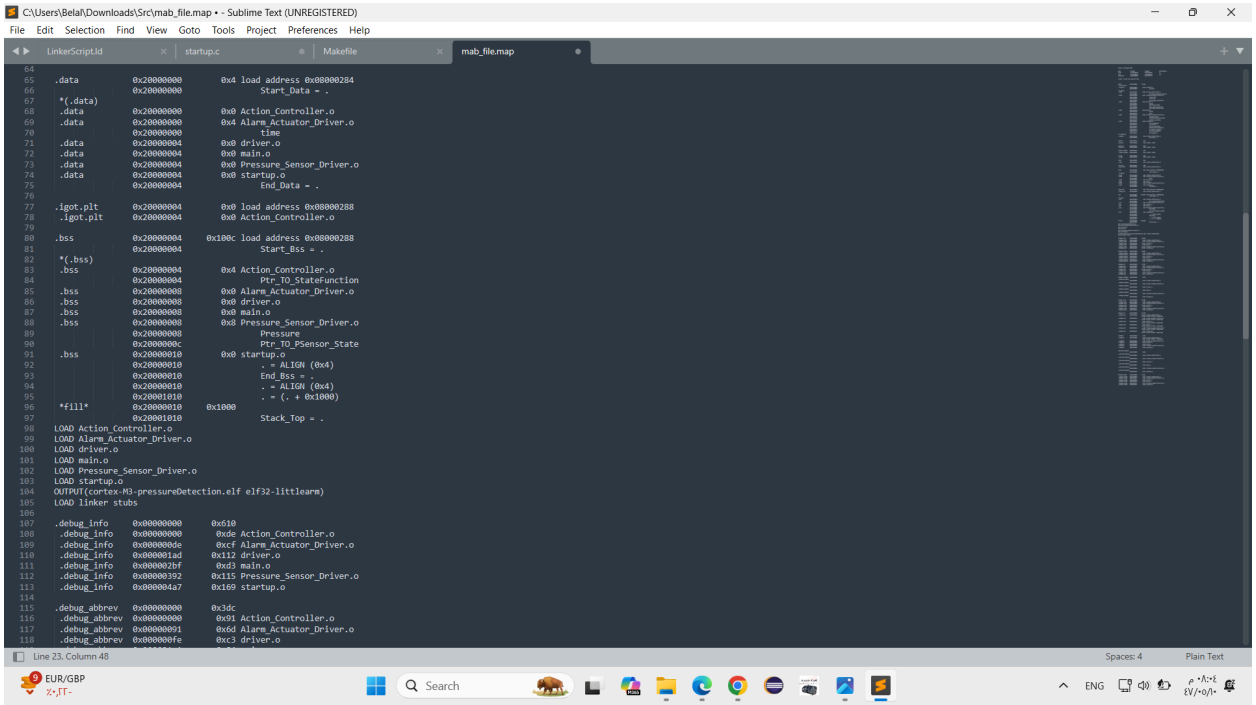

MINGW32:/c/Users/Belal/Downloads/Src
Belal@DESKTOP-O1B5UR8 MINGW32 ~/Downloads/Src
$ arm-none-eabi-nm.exe main.o
U GPIO_INITIALIZATION
00000028 T main
U PSensor_detect_pressure
U PSensor_Read_Pressure
U Ptr_TO_PSensor_State
U Ptr_TO_StateFunction
00000000 T setup


MINGW32:/c/Users/Belal/Downloads/Src
Belal@DESKTOP-O1B5UR8 MINGW32 ~/Downloads/Src
$ 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 00000048 00000000 00000000 00000034 2**2
CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
1 .data 00000000 00000000 00000000 0000007c 2**0
CONTENTS, ALLOC, LOAD, DATA
2 .bss 00000000 00000000 00000000 0000007c 2**0
ALLOC
3 .debug_info 000000d3 00000000 00000000 0000007c 2**0
CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
4 .debug_abbrev 00000084 00000000 00000000 0000014f 2**0
CONTENTS, READONLY, DEBUGGING, OCTETS
5 .debug_loc 00000058 00000000 00000000 000001d3 2**0
CONTENTS, READONLY, DEBUGGING, OCTETS
6 .debug_aranges 00000020 00000000 00000000 0000022b 2**0
CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
7 .debug_line 0000008e 00000000 00000000 0000024b 2**0
CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
8 .debug_str 00000152 00000000 00000000 000002d9 2**0
CONTENTS, READONLY, DEBUGGING, OCTETS
9 .comment 0000004a 00000000 00000000 0000042b 2**0
CONTENTS, READONLY
10 .debug_frame 00000048 00000000 00000000 00000478 2**2
CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
11 .ARM.attributes 0000002d 00000000 00000000 000004c0 2**0
CONTENTS, READONLY

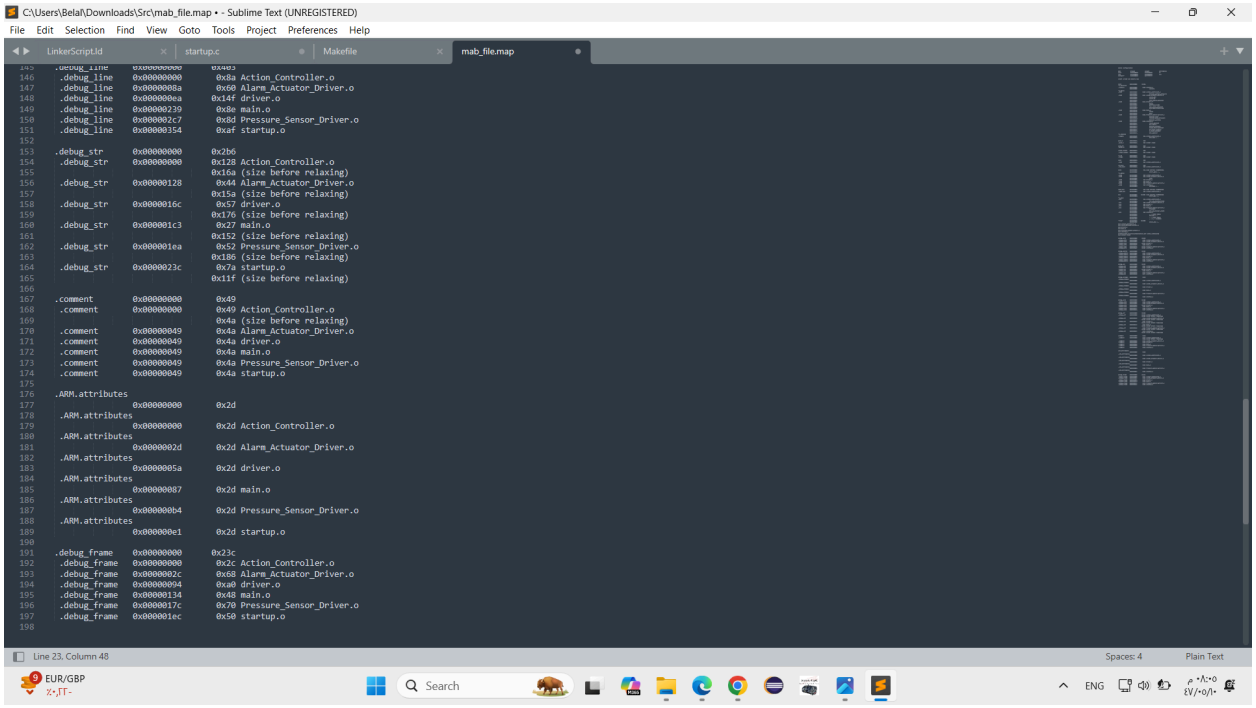
mabfile.map :



```
Memory Configuration

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

Linker script and memory map


.text           0x08000000      0x284
 *(.vectors)
 .vectors       0x08000000       0x20 startup.o
                0x08000000                vectors
 *(.text)
 .text          0x08000020       0x20 Action_Controller.o
                0x08000020                PSensor_detect_pressure
 .text          0x08000040       0x3c Alarm_Actuator_Driver.o
                0x08000040                Alarm_Off
                0x0800004e                Alarm_On
                0x08000070                Init_Alarm_Actuator
 .text          0x0800007c       0xc4 driver.o
                0x0800007c                Delay
                0x0800009e                getPressureVal
                0x080000b4                Set_Alarm_actuator
                0x080000f0                GPIO_INITIALIZATION
 .text          0x08000140       0x48 main.o
                0x08000140                setup
                0x08000168                main
 .text          0x08000188       0x64 Pressure_Sensor_Driver.o
                0x08000188                PSensor_Init
                0x080001a4                PSensor_Read_Pressure
                0x080001d0                PSensor_waiting
 .text          0x080001ec       0x94 startup.o
                0x080001ec                reset_handler
                0x08000274                Bus_fault_
                0x08000274                deafult_handler
                0x08000274                Usage_fault_handler
                0x08000274                MM_fault_handler
                0x08000274                H_fault_handler
                0x08000274                NMI_handler
 *(.rodata)
 .rodata        0x08000280       0x4 Action_Controller.o
                0x08000284                End_Text = .

 .glue_7        0x08000284       0x0
 .glue_7        0x08000284       0x0 linker stubs

 .glue_7t       0x08000284       0x0
 .glue_7t       0x08000284       0x0 linker stubs

 .vfp11_veneer  0x08000284       0x0
 .vfp11_veneer  0x08000284       0x0 linker stubs
```



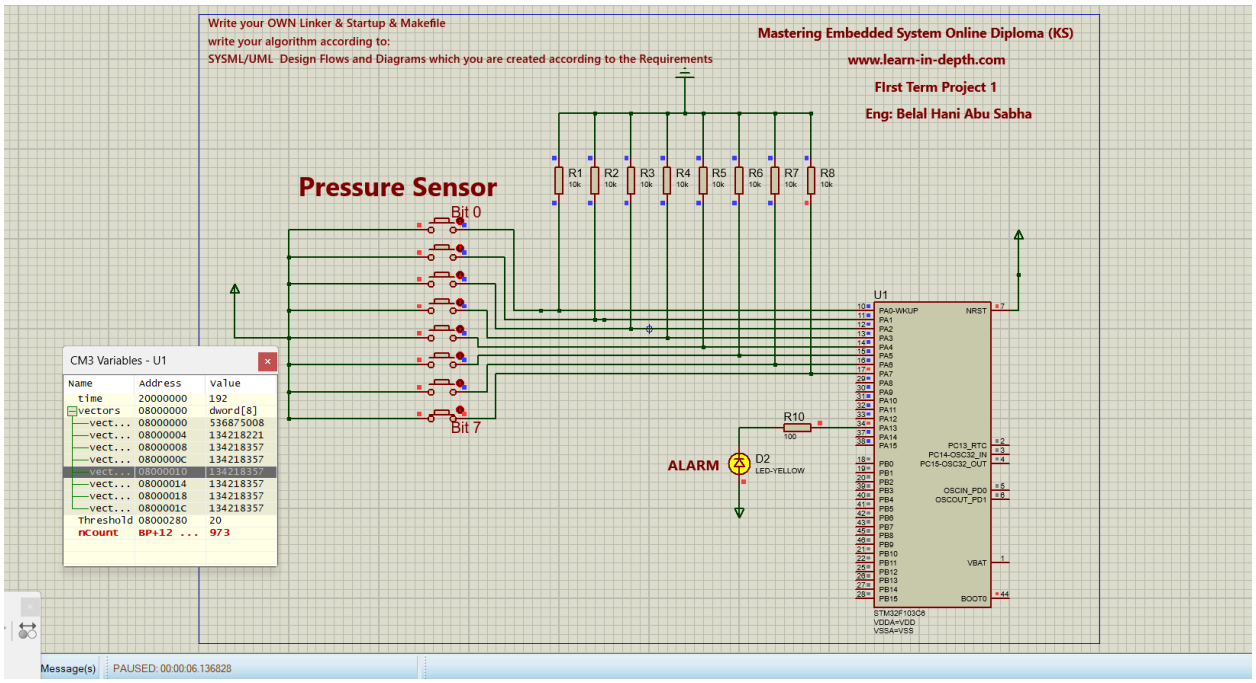
```
 .data          0x20000000       0x4 load address 0x08000284
                0x20000000                Start_Data = .
 *(.data)
 .data          0x20000000       0x0 Action_Controller.o
 .data          0x20000000       0x4 Alarm_Actuator_Driver.o
                0x20000000                time
 .data          0x20000004       0x0 driver.o
 .data          0x20000004       0x0 main.o
 .data          0x20000004       0x0 Pressure_Sensor_Driver.o
 .data          0x20000004       0x0 startup.o
                0x20000004                End_Data = .

 .igot.plt      0x20000004       0x0 load address 0x08000288
 .igot.plt      0x20000004       0x0 Action_Controller.o

 .bss           0x20000004     0x100c load address 0x08000288
                0x20000004                Start_Bss = .
 *(.bss)
 .bss           0x20000004       0x4 Action_Controller.o
                0x20000004                Ptr_TO_StateFunction
 .bss           0x20000008       0x0 Alarm_Actuator_Driver.o
 .bss           0x20000008       0x0 driver.o
 .bss           0x20000008       0x0 main.o
 .bss           0x20000008       0x8 Pressure_Sensor_Driver.o
                0x20000008                Pressure
                0x2000000c                Ptr_TO_PSensor_State
 .bss           0x20000010       0x0 startup.o
                0x20000010                . = ALIGN (0x4)
                0x20000010                End_Bss = .
                0x20000010                . = ALIGN (0x4)
                0x20001010                . = (. + 0x1000)
 *fill*         0x20000010     0x1000
                0x20001010                Stack_Top = .
LOAD Action_Controller.o
LOAD Alarm_Actuator_Driver.o
LOAD driver.o
LOAD main.o
LOAD Pressure_Sensor_Driver.o
LOAD startup.o
OUTPUT(cortex-M3-pressureDetection.elf elf32-littlearm)
LOAD linker stubs

.debug_info     0x00000000      0x610
 .debug_info    0x00000000       0xde Action_Controller.o
 .debug_info    0x000000de       0xcf Alarm_Actuator_Driver.o
 .debug_info    0x000001ad      0x112 driver.o
 .debug_info    0x000002bf       0xd3 main.o
 .debug_info    0x00000392      0x115 Pressure_Sensor_Driver.o
 .debug_info    0x000004a7      0x169 startup.o

.debug_abbrev   0x00000000      0x3dc
 .debug_abbrev  0x00000000       0x91 Action_Controller.o
 .debug_abbrev  0x00000091       0x6d Alarm_Actuator_Driver.o
 .debug_abbrev  0x000000fe       0xc3 driver.o
```


C:\Users\Belal\Downloads\Src\mab_file.map • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
LinkerScript.ld
startup.c
Makefile
mab_file.map
.debug_line 0x00000000 0x8a Action_Controller.o
.debug_line 0x0000008a 0x60 Alarm_Actuator_Driver.o
.debug_line 0x000000ea 0x14f driver.o
.debug_line 0x00000239 0x8e main.o
.debug_line 0x000002c7 0x8d Pressure_Sensor_Driver.o
.debug_line 0x00000354 0xaf startup.o
.debug_str 0x00000000 0x2b6
.debug_str 0x00000000 0x128 Action_Controller.o
0x16a (size before relaxing)
.debug_str 0x00000128 0x44 Alarm_Actuator_Driver.o
0x15a (size before relaxing)
.debug_str 0x0000016c 0x57 driver.o
0x176 (size before relaxing)
.debug_str 0x000001c3 0x27 main.o
0x152 (size before relaxing)
.debug_str 0x000001ea 0x52 Pressure_Sensor_Driver.o
0x186 (size before relaxing)
.debug_str 0x0000023c 0x7a startup.o
0x11f (size before relaxing)
.comment 0x00000000 0x49
.comment 0x00000000 0x49 Action_Controller.o
0x4a (size before relaxing)
.comment 0x00000049 0x4a Alarm_Actuator_Driver.o
.comment 0x00000049 0x4a driver.o
.comment 0x00000049 0x4a main.o
.comment 0x00000049 0x4a Pressure_Sensor_Driver.o
.comment 0x00000049 0x4a startup.o
.ARM.attributes
0x00000000 0x2d
.ARM.attributes
0x00000000 0x2d Action_Controller.o
.ARM.attributes
0x0000002d 0x2d Alarm_Actuator_Driver.o
.ARM.attributes
0x0000005a 0x2d driver.o
.ARM.attributes
0x00000087 0x2d main.o
.ARM.attributes
0x000000b4 0x2d Pressure_Sensor_Driver.o
.ARM.attributes
0x000000e1 0x2d startup.o
.debug_frame 0x00000000 0x23c
.debug_frame 0x00000000 0x2c Action_Controller.o
.debug_frame 0x0000002c 0x68 Alarm_Actuator_Driver.o
.debug_frame 0x00000094 0xa0 driver.o
.debug_frame 0x00000134 0x48 main.o
.debug_frame 0x0000017c 0x70 Pressure_Sensor_Driver.o
.debug_frame 0x000001ec 0x50 startup.o
Line 23, Column 48
Spaces: 4
Plain Text
EUR/GBP
Search
ENG

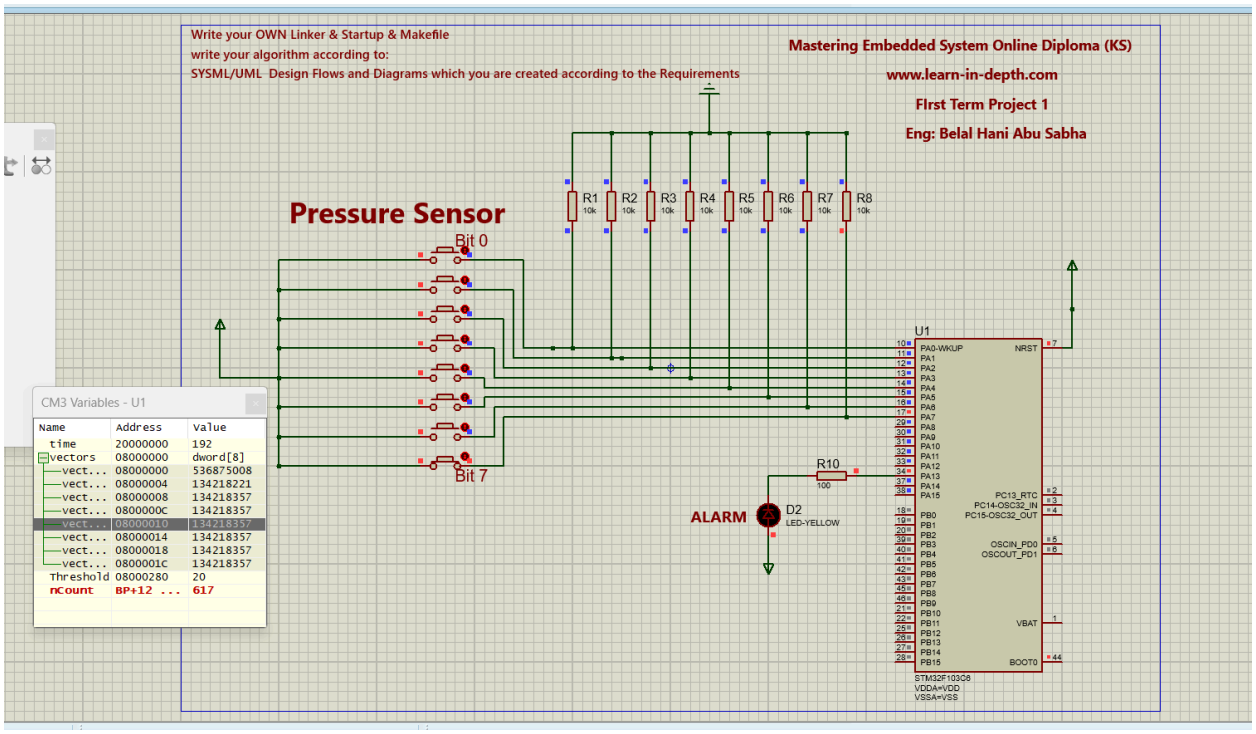
Simulation Using Proteus :

-if Pressure above or equal 20 bar then Alarm will turn on for 60 seconds

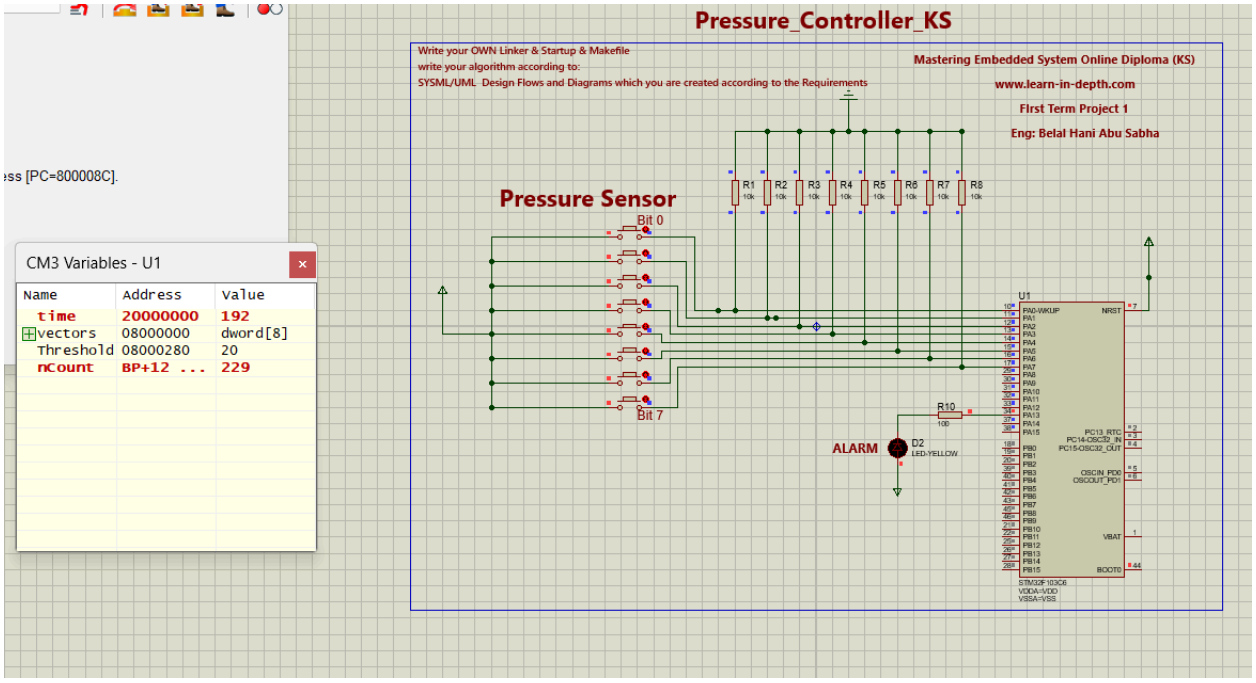
.



Alarm is off after 60 seconds :



else :


ss [PC=800008C].
CM3 Variables - U1
Name
Address
Value
time
20000000
192
Vectors
08000000
dword[8]
Threshold
08000280
20
nCount
BP+12 ...
229
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: Belal Hani Abu Sabha
Pressure Sensor
Bit 0
Bit 7
R1
R2
R3
R4
R5
R6
R7
R8
R10
ALARM
D2
LED-YELLOW
U1

Conclusion :

Results:

1-Successfully implemented pressure detection system on STM32F103C6

2-System automatically senses pressure and controls alarm

3-All functional requirements achieved

Future Improvements:

1-Store pressure value in Memory