Mastering Embedded
System Online Diploma

First Term
[Final Project 1]

Pressure Controller

www.learn-in-depth.com [By Eng.K.S.]

Eng. Mohamed Abd El Hakeem El Said Ali.

My Profile:
eng.mohamedhakim2122000@gmail.com (learn-in-depth-store.com)

1 First Stage [System Architecting]:

1 Case Study: [With Customer]
2 Method
3 Requirement
4 Space Exploration
5 System Analysis
6 System Design

1 Case Study: [From Customer]

- A" client" expects you to deliver the software of the following system: -
  - Specification (from the client) :
    1. A pressure controller informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin.
    2. The alarm duration equals 60 seconds.
    3. keeps track of the measured values.

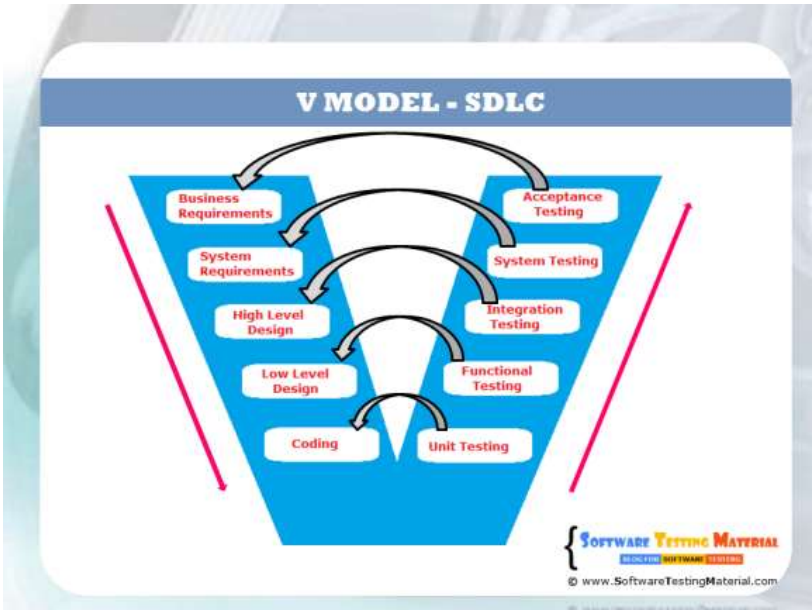
-My Assumptions:

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 a power cut.

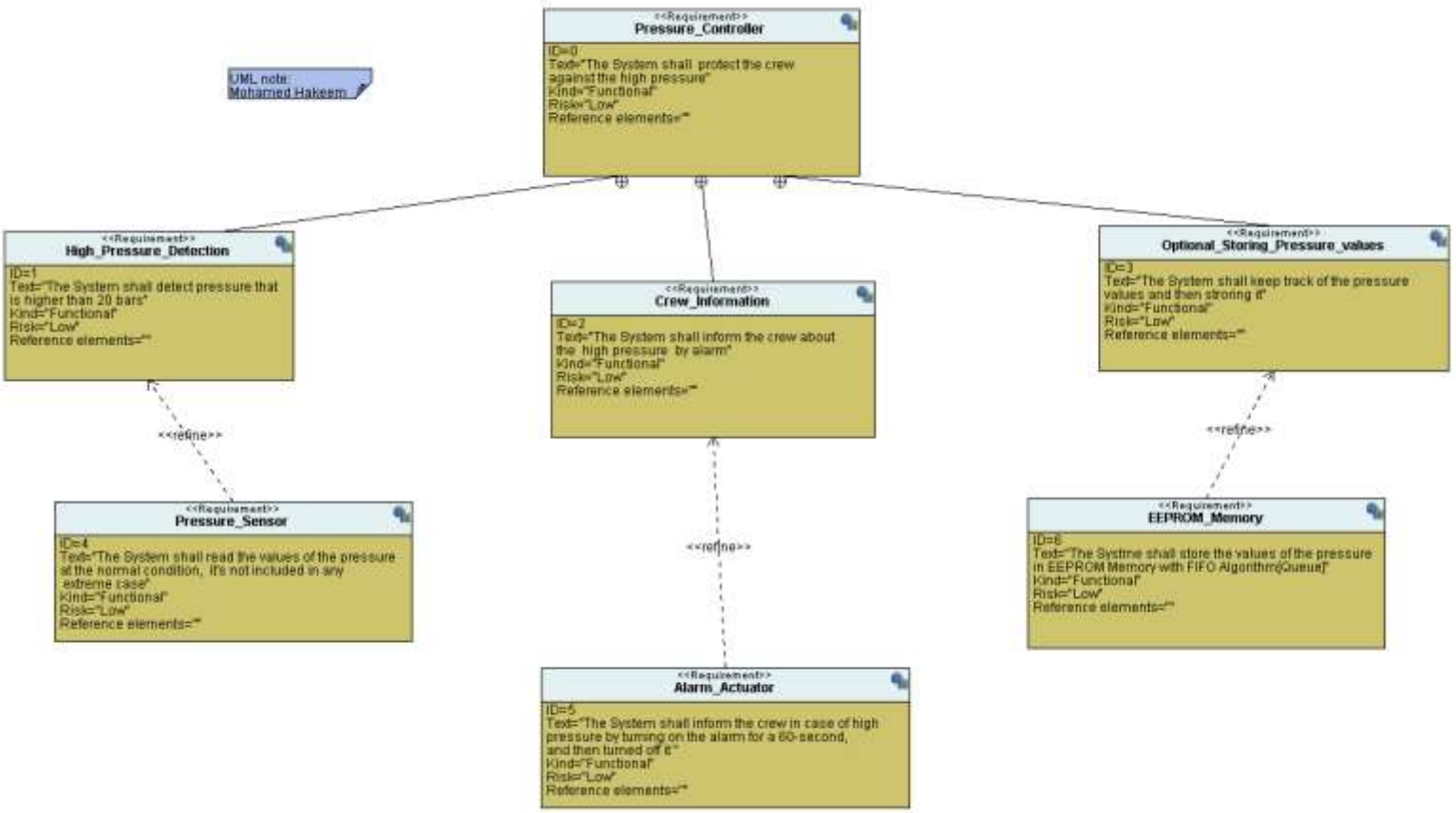
- Versioning • The "keep track of measured value" option is not modeled in the first version of the design

2 Method:

- I would work with the Vmodel method to apply my system design.

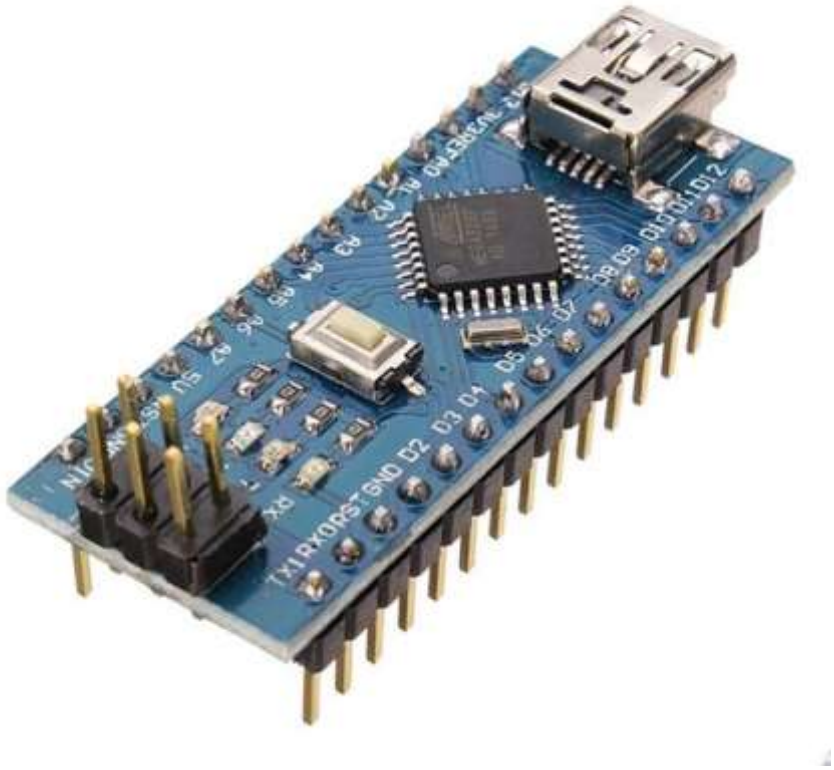


3 Requirements:



4 Space Exploration:

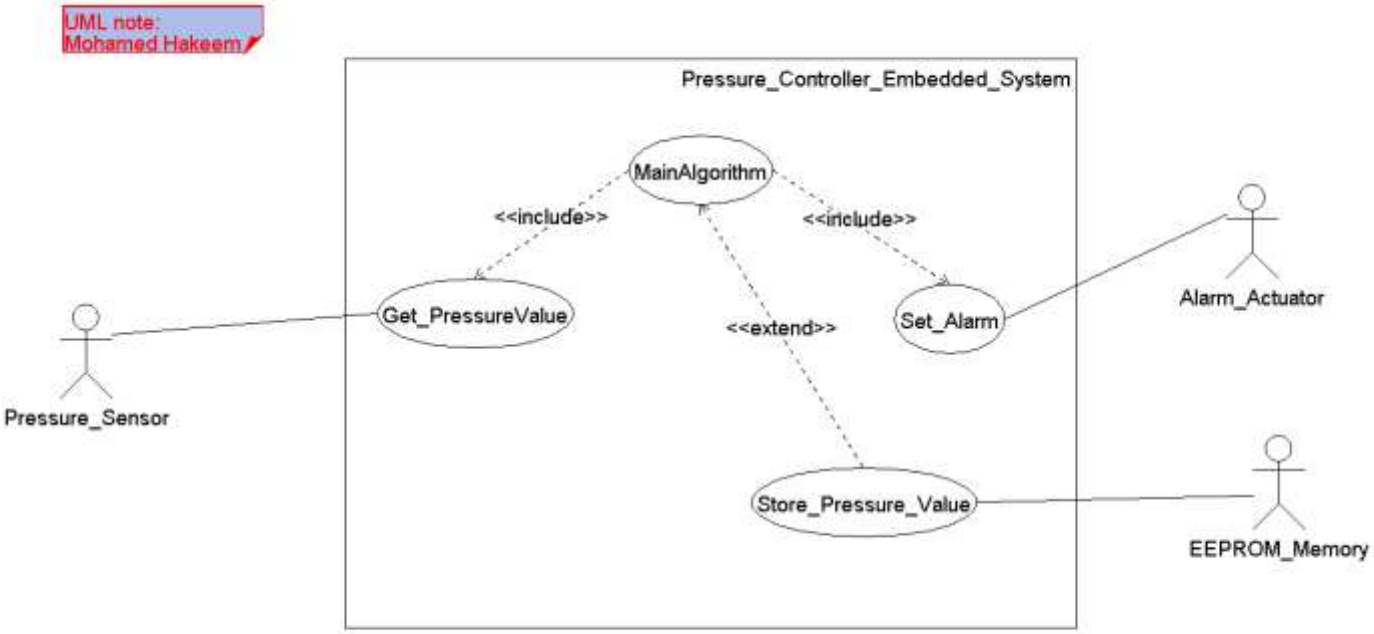
-I see that Board "STM32F103C8T6" "Cortex_M3" is Very suitable for this Case study.



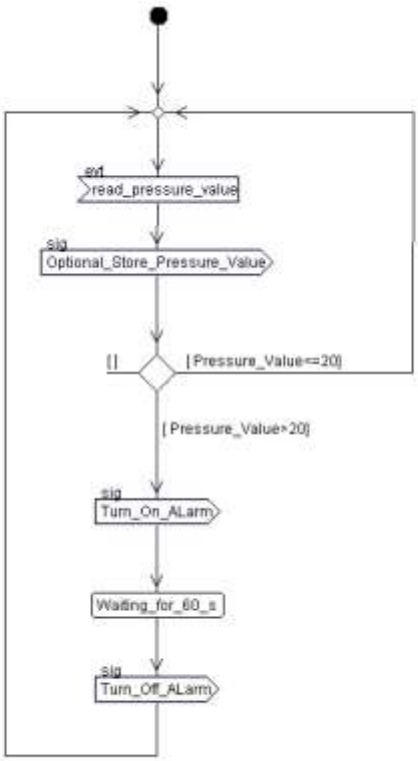
- Core: 32-bit ARM Cortex-M3 CPU, running at up to 72 MHz
- Memory: 64 KB of Flash memory and 20 KB of SRAM.
- Peripherals: Includes two 12-bit ADCs, three general-purpose 16-bit timers, one PWM timer, and various communication interfaces like I2C, SPI, USART, USB, and CAN.
- Power: Operates from a 2.0 to 3.6 V power supply and includes several power-saving modes.

5 System Analysis:

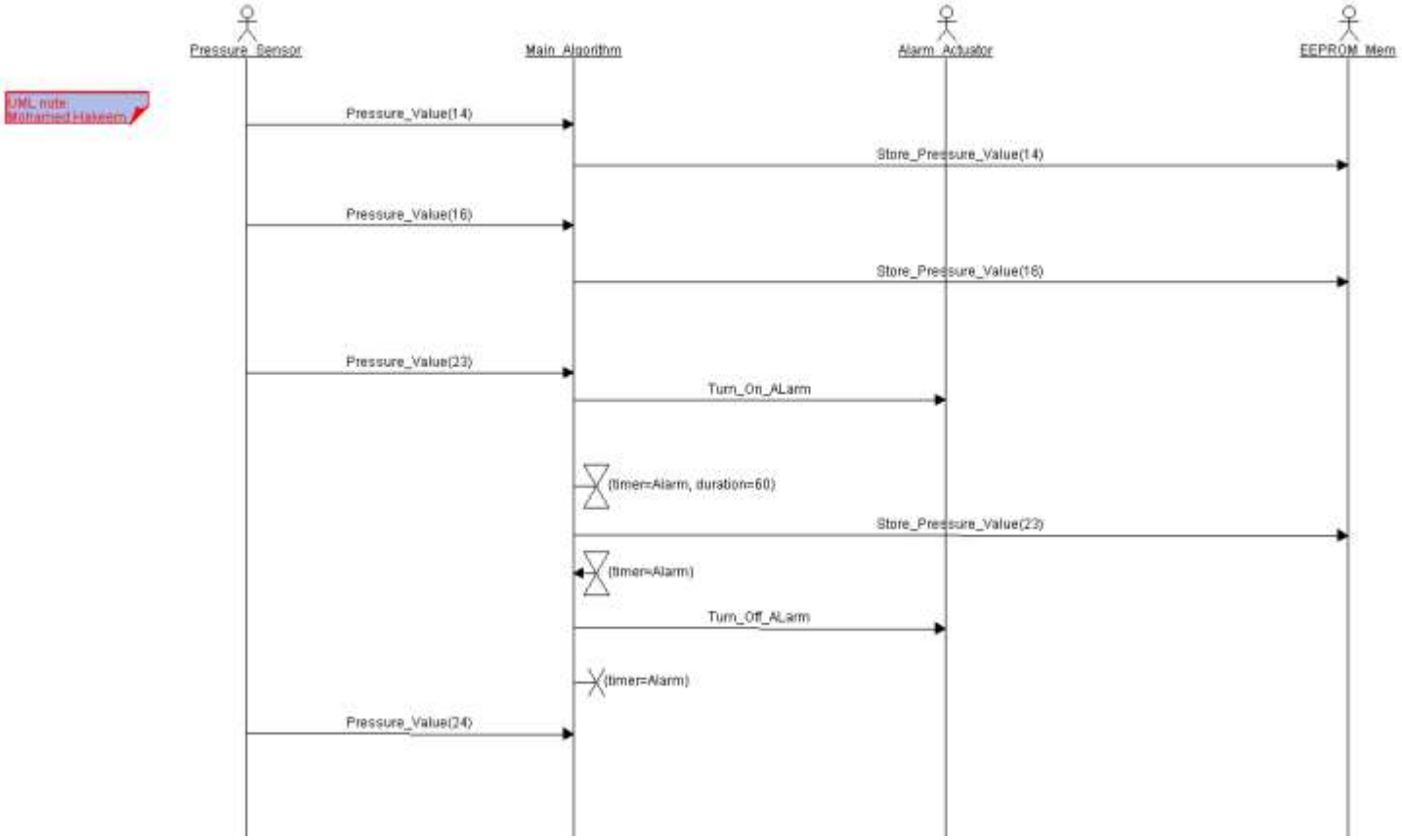
1 Use Case:



2 Activity Diagram:

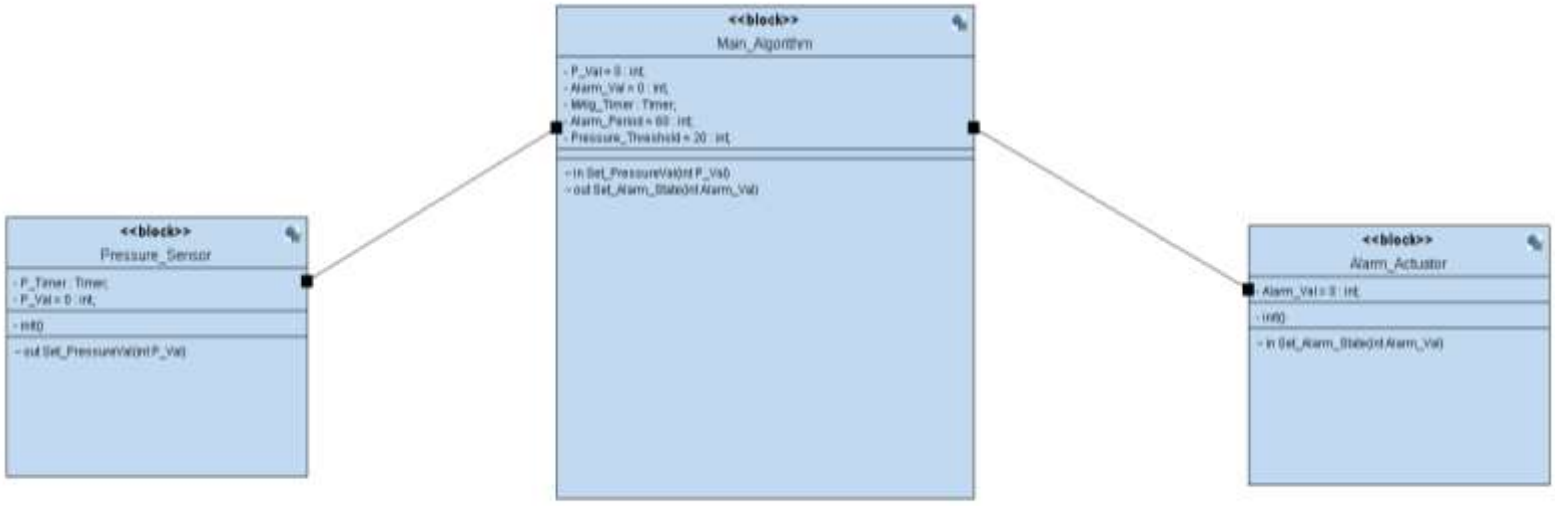


3 Sequence Diagram:

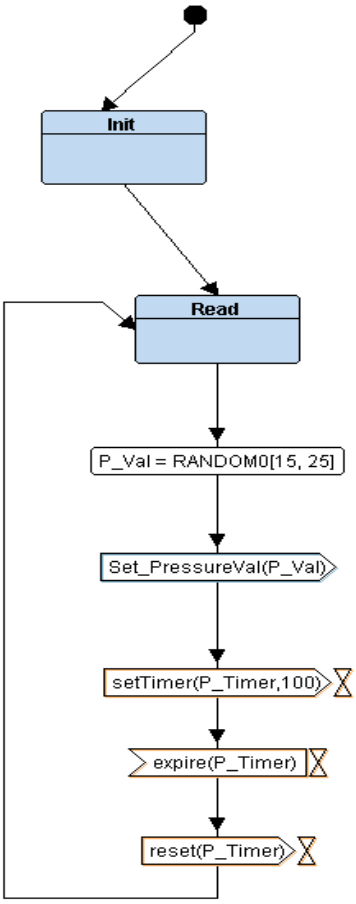


6 System Design:

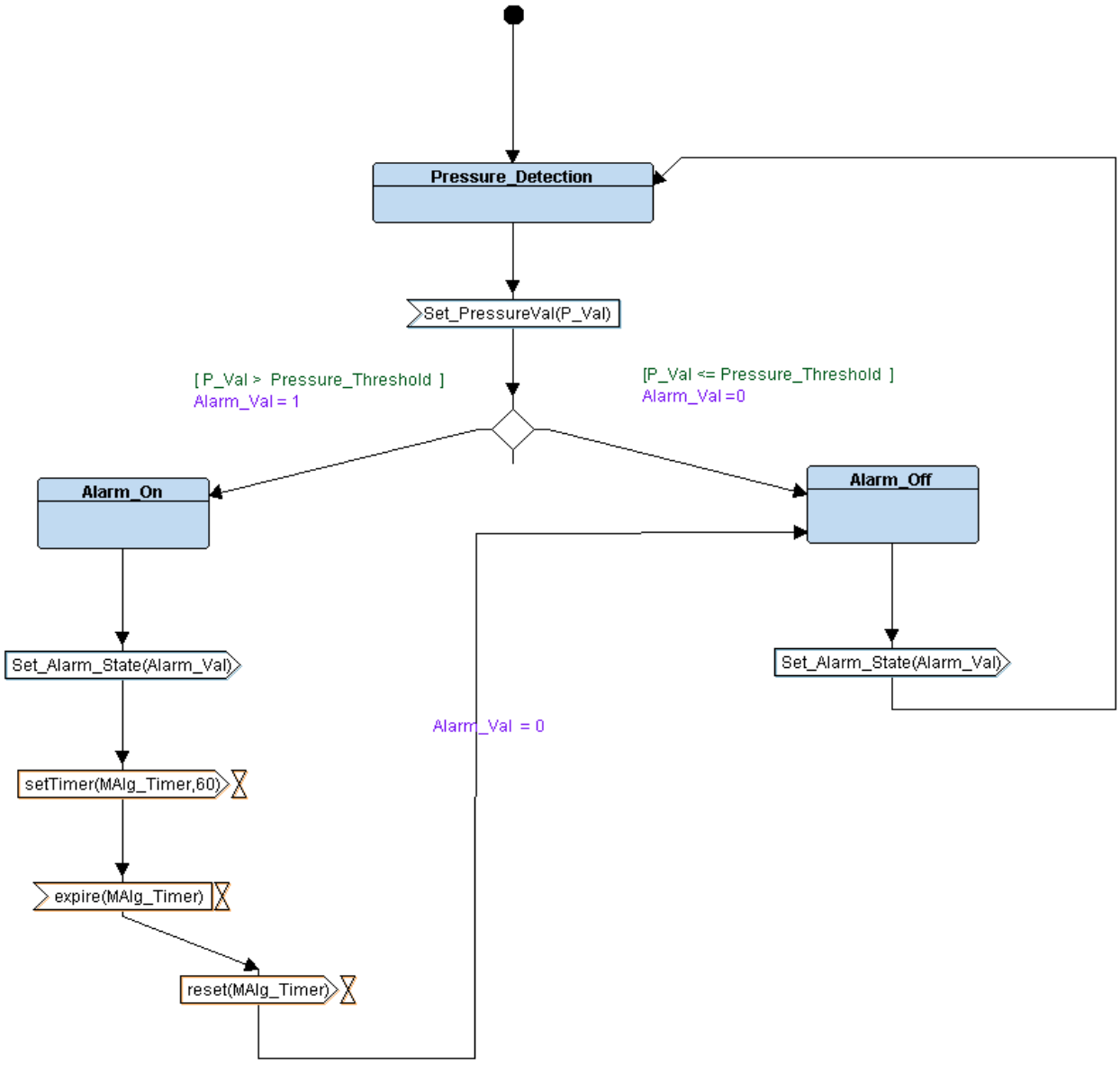
1 Block Diagram:



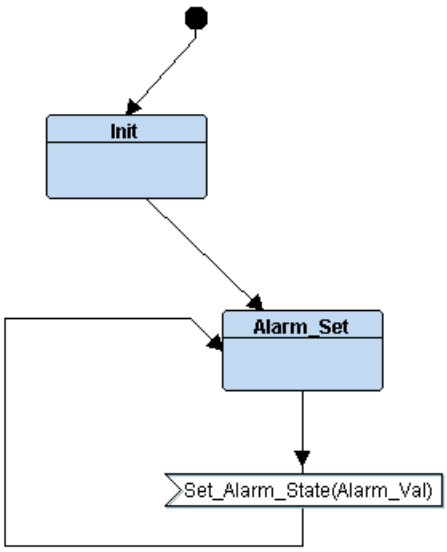
2 Pressure Sensor:



3 Main Algorithm:



4 Alarm Actuator:



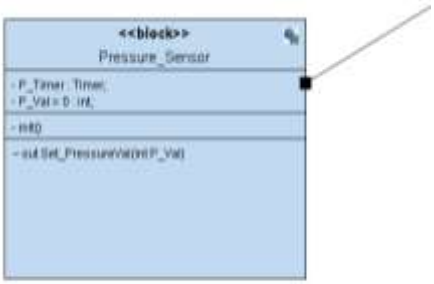
2 Second Stage [Embedded Implementation]:

-State.h:

```c
/*
 * state.h
 *
 *  Created on: Aug 19, 2024
 *      Author: Eng:Mohamed Abd El Hakeem El Said Ali
 */

#ifndef STATE_H_
#define STATE_H_


#include <stdio.h>
#include <stdlib.h>

/*Factory of function for the tasks*/
#define State_Declare(State_Fun)    void State_Fun()
#define State_Add(State_Fun)  State_Fun







/*Functions to communicate between the task and each other*/
//Map between the tasks "Pressure_Sensor " & "MainALgorithm" :using this func in MainAlgorithm
void Set_PressureVal(int P_Val);
//Map between the tasks "MainALgorithm" & "Alarm_Actuator"  :using this func in Alarm_Actuator
void Set_Alarm_State(int Alarm_Val);


#endif /* STATE_H_ */

```

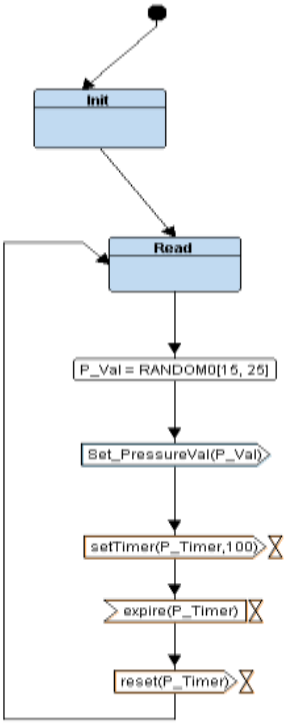
1 Pressure Sensor.c:

```c
/*
 * Pressure_Sensor.c
 *
 *  Created on: Aug 19, 2024
 *      Author: Eng:Mohamed Abd El Hakeem El Said Ali
 *
 *  [Pressure_Sensor] is task or file that responsible for
 *  to the task is called  [Main_Algorithm] that is task th
 */


#include "Pressure_Sensor.h"
unsigned int P_Val=0;
extern void (*PS_State)();




State_Declare(PS_init)
{

    PS_State_ID=PS_Init;



}
State_Declare(PS_read)
{

    PS_State_ID=PS_Read;
    P_Val=getPressureVal();

    Set_PressureVal(P_Val);


    /*Go back to the same state 'PS_read'*/
    PS_State=State_Add(PS_read);


}
```



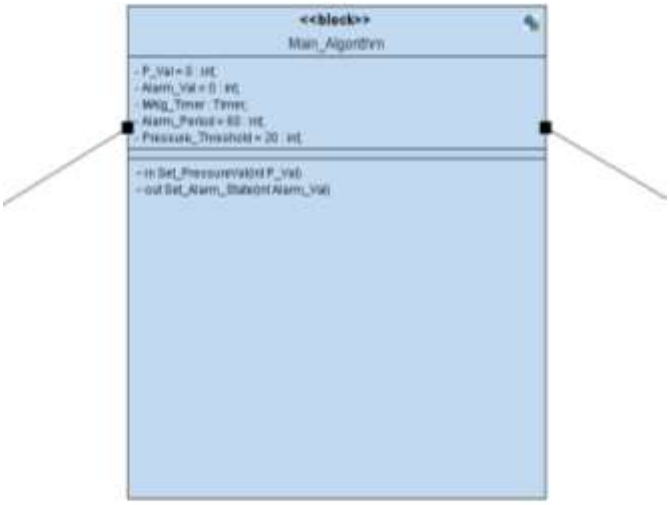
2 Pressure Sensor.h:

```c
/*
 * Pressure_Sensor.h
 *
 *  Created on: Aug 19, 2024
 *      Author: Eng:Mohamed Abd El Hakeem El Said Ali
 */

#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_

#include"state.h"

int generate_random(int l,int r,int count);


/*states that you made in the tool of state diagram*/
enum
{
    PS_Init,
    PS_Read,

}PS_State_ID;


State_Declare(PS_init);
State_Declare(PS_read);


//global pointer to func;
void (*PS_State)();


#endif /* PRESSURE_SENSOR_H_ */
```



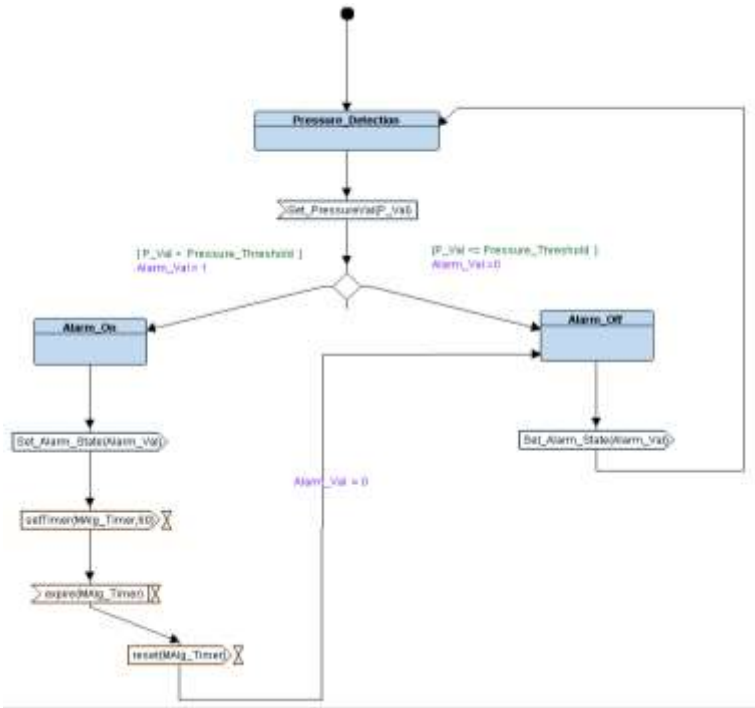
3 Main_Algorithm.c:

```c
#include "Main_Algorithm.h"

unsigned int P_Value,Alarm_Val,Alarm_Period=60,Pressure_Threshold=20;
extern void (*MA_State)();


State_Declare(MA_pressureDetection)
{
    MA_State_ID=MA_PressureDetection;
    MA_State=Set_PressureVal;

}
void Set_PressureVal(int P_Val)
{

    P_Value=P_Val;
    (P_Value<=Pressure_Threshold)?(MA_State=State_Add(MA_alarmOff)):(MA_State=State_Add(MA_alarmOn));

}
State_Declare(MA_alarmOff)
{
    MA_State_ID=MA_AlarmOff;

    /*Value of 1 is Off in Simulation ,so I made it as this*/
    Alarm_Val=1;
    Set_Alarm_State(Alarm_Val);

    MA_State=State_Add(MA_pressureDetection);

}



State_Declare(MA_alarmOn)
{
    MA_State_ID=MA_AlarmOn;

    /*Value of 0 is 1 in Simulation ,so I made it as this*/
    Alarm_Val=0;
    Set_Alarm_State(Alarm_Val);


    MA_State=State_Add(MA_alarmOff);



}
```



4 Main_Algorithm.h:

```c
/*
 *
 *  Main_Algorithm.h
 *
 *  Created on: Aug 19, 2024
 *      Author: Eng:Mohamed Abd El Hakeem El Said Ali
 *      Main_Algorithm task is the algorithm that control between the another tasks
 */

#ifndef MAIN_ALGORITHM_H_
#define MAIN_ALGORITHM_H_

#include"state.h"


/*states that you made in the tool of state diagram*/
enum
{
    MA_PressureDetection,
    MA_AlarmOn,
    MA_AlarmOff,

}MA_State_ID;


State_Declare(MA_pressureDetection);
State_Declare(MA_alarmOn);
State_Declare(MA_alarmOff);



//global pointer to func;
void (*MA_State)();


#endif /* MAIN_ALGORITHM_H_ */
```

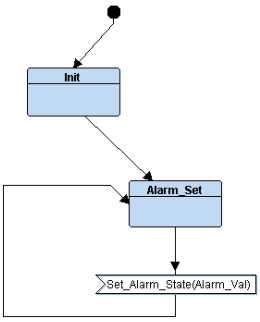


5 Alarm_Actuator.c:

```c
/*
 * Alarm_Actuator.c
 *
 *  Created on: Aug 19, 2024
 *      Author: Eng:Mohamed Abd El Hakeem El Said Ali
 *
 *
 *      [Alarm_Actuator] is the task that receives actor(Alarm value) fro
 */

#include "Alarm_Actuator.h"
unsigned int Alarm_Value=0;

extern void (*AL_State)();


void Set_Alarm_State(int Alarm_Val)
{


    Alarm_Value=Alarm_Val;
    AL_State=State_Add(AL_alarmSet);



}

State_Declare(AL_init)
{

    AL_State_ID=AL_Init;

}

State_Declare(AL_alarmSet)
{
    AL_State_ID=AL_AlarmSet;
    //send Alarm_Value to Alarm_Actuator
    Set_Alarm_actuator(Alarm_Value);
    AL_State=State_Add(AL_alarmSet);

}
```

6 Alarm_Actuator.h:

```c
/*
 * Alarm_Actuator.h
 *
 *  Created on: Aug 19, 2024
 *      Author: Eng:Mohamed Abd El Hakeem El Said Ali
 *
 */

#ifndef ALARM_ACTUATOR_H_
#define ALARM_ACTUATOR_H_


#include"state.h"

/*states that you made in the tool of state diagram*/
enum
{
    AL_Init,
    AL_AlarmSet,

}AL_State_ID;

State_Declare(AL_init);
State_Declare(AL_alarmSet);

//global pointer to func;
void (*AL_State)();



#endif /* ALARM_ACTUATOR_H_ */
```

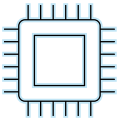


7 main.c:

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


#include "Alarm_Actuator.h"
#include "Main_Algorithm.h"
#include "Pressure_Sensor.h"


void setup_app()
{

    GPIO_INITIALIZATION();
    State_Declare(AL_init);
    State_Declare(PS_init);

    //set states pointers for each first task states
    PS_State=State_Add(PS_read);
    MA_State=State_Add(MA_pressureDetection);
    AL_State=State_Add(AL_alarmSet);

}


int main()
{

    setup_app();


    while(1)
    {
        //call the pointer to function respectively according to your state diagram
        PS_State();
        MA_State();
        AL_State();
        if(MA_State_ID==MA_AlarmOn)
        {
            /*Set timer to 60 s for waiting the turning on of alarm*/
            Delay(2000000);
            MA_State();
            AL_State();
        }
        else
        {
            //Do Nothing
        }
    }
    return 0;
}
```

8 driver.c:

```c
/*
 * driver.c
 *
 *  Created on: Aug 19, 2024
 *      Author: Eng:Mohamed Abd El Hakeem El Said Ali
 *
 */


#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;

    SET_BIT(GPIOA_ODR,13);

}
```



9 Startup.c:

```c
/*learn-in-depth u
  Mohamed Abd El Hakeem EL Said Ali.
  startup.c
  */

/*SRAM 0x20000000 */


#include<stdint.h>
#include"Platform_Types.h"
extern int main(void);

void Reset_Handler(void);
void Default_Handler(void);


void NMI_Handler(void) __attribute__((weak,alias("Default_Handler")));
void Hard_Fault_Handler(void)__attribute__((weak,alias("Default_Handler")));
void MM_Fault_Handler(void)__attribute__((weak,alias("Default_Handler")));
void Bus_Fault_Handler(void)__attribute__((weak,alias("Default_Handler")));
void Usage_Fault_Handler(void)__attribute__((weak,alias("Default_Handler")));


extern uint32 Stack_Top;

uint32 Vectors[] __attribute__((section(".vectors")))={

(uint32) &Stack_Top,
(uint32) &Reset_Handler,
(uint32) &NMI_Handler,
(uint32) &Hard_Fault_Handler,
(uint32) &MM_Fault_Handler,
(uint32) &Bus_Fault_Handler,
(uint32) &Usage_Fault_Handler,

};

extern uint32 E_text;
extern uint32 S_data;
extern uint32 E_data;
extern uint32 S_bss;
extern uint32 E_bss;
int i;

void Reset_Handler(void)
{
  //Copy data section from flash to ram
  uint32 Data_Size=(uint8*)&E_data-(uint8*)&S_data;
  uint8* Ptr_Src=(uint8*)&E_text;
  uint8* Ptr_Dst=(uint8*)&S_data;


  for(i=0;i<Data_Size;i++)
  {
```

```c
void Reset_Handler(void)
{
  //Copy data section from flash to ram
  uint32 Data_Size=(uint8*)&E_data-(uint8*)&S_data;
  uint8* Ptr_Src=(uint8*)&E_text;
  uint8* Ptr_Dst=(uint8*)&S_data;


  for(i=0;i<Data_Size;i++)
  {

    *((uint8*)Ptr_Dst++)=*((uint8*)Ptr_Src++);
  }

  //Initialization bss section in SRAM with Zero
  uint32 bss_Size=(uint8*)&E_bss-(uint8*)&S_bss;
  Ptr_Dst=(uint8*)&S_bss;


  for(i=0;i<Data_Size;i++)
  {

    *((uint8*)Ptr_Dst++)=((uint8) 0 );
  }

  main();

}

void Default_Handler(void)
{
  Reset_Handler();
}
```

10 Linker_Script.ld:

```
/*learn-in-depth
  Mohamed Abd El Hakeem EL Said Ali.
  linker_script.ld
*/

MEMORY
{

    flash(RX) : ORIGIN = 0x08000000, LENGTH = 128K
    sram(RWX) : ORIGIN = 0x20000000, LENGTH = 20K

}

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

  .data :
  {
    S_data = . ;
    *(.data)
    E_data = . ;

  } > sram AT> flash

  .bss :
   {

    S_bss = . ;
    *(.bss*)
    E_bss = . ;
    . = . + 0x1000;
    Stack_Top = . ;


    } > sram
}
```

11 Makefile:

```
#Copyright: Mohamed Abd El Hakeem El Said El Ali .
CC=arm-none-eabi-
CFLAGES= -gdwarf-2 -mcpu=cortex-m3 -mthumb   #gdwarf-2 is defined to protuse debugger  & -mthumb to operate on my  target cpu
INCS=-I .                   #Add Paths of Header Files ---> . mean same folder.
LIBS=                       #For including static library 'files.a' or'files.lib' in Linking time.
SRC=$(wildcard *.c)         #wildcard searchd any files.c   in the folder.
OBJ=$(SRC:.c=.o)
AS=$(wildcard *.s)          #wildcard searchd any files.s   in the folder.
ASOBJ=$(AS:.s=.o)
Project_Name=Pressure_Controller

all:$(Project_Name).bin
    @echo "=============Build is Done================"

%.o:%.s
    $(CC)as.exe  $(CFLAGES)  $< -o $@

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

$(Project_Name).elf:$(OBJ) $(ASOBJ)
    $(CC)ld.exe  -T linker_script.ld $(LIBS) $(OBJ) $(ASOBJ) -o $@ -Map=Mapfile.map

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


clear_all:
    rm *.o *.elf *.bin *.map

clear:
    rm *.elf  *.bin
```

12-Building By tool chain[arm-none-eabi-x]:

```
PC-SHOP@DESKTOP-TSGOOR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (m
$ make all
arm-none-eabi-gcc.exe -c -I .                   -gdwarf-2 -mcpu=cortex-m3 -mthumb     Alarm_Actuator.c -o Alarm_Actuator.o
arm-none-eabi-gcc.exe -c -I .                   -gdwarf-2 -mcpu=cortex-m3 -mthumb     driver.c -o driver.o
arm-none-eabi-gcc.exe -c -I .                   -gdwarf-2 -mcpu=cortex-m3 -mthumb     main.c -o main.o
arm-none-eabi-gcc.exe -c -I .                   -gdwarf-2 -mcpu=cortex-m3 -mthumb     Main_Algorithm.c -o Main_Algorithm.o
arm-none-eabi-gcc.exe -c -I .                   -gdwarf-2 -mcpu=cortex-m3 -mthumb     Pressure_Sensor.c -o Pressure_Sensor.o
arm-none-eabi-gcc.exe -c -I .                   -gdwarf-2 -mcpu=cortex-m3 -mthumb     startup.c -o startup.o
arm-none-eabi-ld.exe  -T linker_script.ld  Alarm_Actuator.o driver.o main.o Main_Algorithm.o Pressure_Sensor.o startup.o  -o Pressure_Controller.elf -Map=Mapfi
arm-none-eabi-objcopy.exe -O binary Pressure_Controller.elf Pressure_Controller.bin
==============Build is Done==================
```

13-Mapfile.map:

```
Allocating common symbols
Common symbol       size              file

AL_State_ID         0x1               Alarm_Actuator.o
P_Value             0x4               Main_Algorithm.o
MA_State            0x4               main.o
Alarm_Val           0x4               Main_Algorithm.o
i                   0x4               startup.o
AL_State            0x4               Alarm_Actuator.o
PS_State            0x4               main.o
MA_State_ID         0x1               main.o
PS_State_ID         0x1               main.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      0x4d4
 *(.vectors*)
 .vectors       0x08000000      0x1c startup.o
                0x08000000              Vectors
 *(.text*)
 .text          0x0800001c      0x80 Alarm_Actuator.o
                0x0800001c              Set_Alarm_State
                0x0800004c              AL_init
                0x08000064              AL_alarmSet
 .text          0x0800009c      0x124 driver.o
                0x0800009c              Delay
                0x080000c8              getPressureVal
                0x080000d8              Set_Alarm_actuator
                0x08000128              GPIO_INITIALIZATION
 .text          0x080001c0      0xa4 main.o
                0x080001c0              setup_app
                0x08000200              main
 .text          0x08000264      0x110 Main_Algorithm.o
                0x08000264              MA_pressureDetection
                0x08000290              Set_PressureVal
                0x080002ec              MA_alarmOff
                0x08000330              MA_alarmOn
 .text          0x08000374      0x60 Pressure_Sensor.o
                0x08000374              PS_init
                0x0800038c              PS_read
 .text          0x080003d4      0x100 startup.o
                0x080003d4              Reset_Handler
                0x080004c8              MM_Fault_Handler
                0x080004c8              Bus_Fault_Handler
                0x080004c8              Default_Handler
                0x080004c8              Usage_Fault_Handler
                0x080004c8              Hard_Fault_Handler
```

```
 *(.rodata)
                0x080004d4                E_text = .

.glue_7         0x080004d4        0x0
 .glue_7        0x00000000        0x0 linker stubs

.glue_7t        0x080004d4        0x0
 .glue_7t       0x00000000        0x0 linker stubs

.vfp11_veneer   0x080004d4        0x0
 .vfp11_veneer  0x00000000        0x0 linker stubs

.v4_bx          0x080004d4        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x080004d4        0x0
 .iplt          0x00000000        0x0 Alarm_Actuator.o

.rel.dyn        0x080004d4        0x0
 .rel.iplt      0x00000000        0x0 Alarm_Actuator.o

.data           0x20000000        0x8 load address 0x080004d4
                0x20000000                S_data = .
 *(.data)
 .data          0x20000000        0x0 Alarm_Actuator.o
 .data          0x20000000        0x0 driver.o
 .data          0x20000000        0x0 main.o
 .data          0x20000000        0x8 Main_Algorithm.o
                0x20000000                Alarm_Period
                0x20000004                Pressure_Threshold
 .data          0x20000008        0x0 Pressure_Sensor.o
 .data          0x20000008        0x0 startup.o
                0x20000008                E_data = .

.igot.plt       0x20000008        0x0 load address 0x080004dc
 .igot.plt      0x00000000        0x0 Alarm_Actuator.o

.bss            0x20000008     0x1028 load address 0x080004dc
                0x20000008                S_bss = .
 *(.bss*)
 .bss           0x20000008        0x4 Alarm_Actuator.o
                0x20000008                Alarm_Value
 .bss           0x2000000c        0x0 driver.o
 .bss           0x2000000c        0x0 main.o
 .bss           0x2000000c        0x0 Main_Algorithm.o
 .bss           0x2000000c        0x4 Pressure_Sensor.o
                0x2000000c                P_Val
 .bss           0x20000010        0x0 startup.o
                0x20000010                E_bss = .
                0x20001010                . = (. + 0x1000)
 *fill*         0x20000010     0x1000
                0x20001010                Stack_Top = .
 COMMON         0x20001010        0x8 Alarm_Actuator.o
                0x20001010                AL_State_ID
                0x20001014                AL_State
```

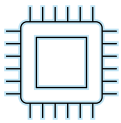
```
 *fill*         0x20001022        0x2
 COMMON         0x20001024        0x8 Main_Algorithm.o
                0x20001024                P_Value
                0x20001028                Alarm_Val
 COMMON         0x2000102c        0x4 startup.o
                0x2000102c                i
LOAD Alarm_Actuator.o
LOAD driver.o
LOAD main.o
LOAD Main_Algorithm.o
LOAD Pressure_Sensor.o
LOAD startup.o
OUTPUT(Pressure_Controller.elf elf32-littlearm)
```

14-Symbol Tables:

```
PC-SHOP@DESKTOP-TSGODR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-nm Pressure_Sensor.o
         U getPressureVal
00000000 B P_Val
00000000 T PS_init
00000018 T PS_read
00000004 C PS_State
00000001 C PS_State_ID
         U Set_PressureVal

PC-SHOP@DESKTOP-TSGODR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-nm Main_Algorithm.o
00000000 D Alarm_Period
00000004 C Alarm_Val
00000088 T MA_alarmOff
000000cc T MA_alarmOn
00000000 T MA_pressureDetection
00000004 C MA_State
00000001 C MA_State_ID
00000004 C P_Value
00000004 D Pressure_Threshold
         U Set_Alarm_State
0000002c T Set_PressureVal

PC-SHOP@DESKTOP-TSGODR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-nm Alarm_Actuator.o
00000048 T AL_alarmSet
00000030 T AL_init
00000004 C AL_State
00000001 C AL_State_ID
00000000 B Alarm_Value
         U Set_Alarm_actuator
00000000 T Set_Alarm_State
```

```
PC-SHOP@DESKTOP-TSGODR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-nm main.o
         U AL_alarmSet
00000004 C AL_State
00000001 C AL_State_ID
         U Delay
         U GPIO_INITIALIZATION
         U MA_pressureDetection
00000004 C MA_State
00000001 C MA_State_ID
00000040 T main
         U PS_read
00000004 C PS_State
00000001 C PS_State_ID
00000000 T setup_app

PC-SHOP@DESKTOP-TSGODR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-nm driver.o
00000000 T Delay
00000024 T getPressureVal
0000008c T GPIO_INITIALIZATION
0000003c T Set_Alarm_actuator

PC-SHOP@DESKTOP-TSGODR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-nm startup.o
000000f4 W Bus_Fault_Handler
000000f4 T Default_Handler
         U E_bss
         U E_data
         U E_text
000000f4 W Hard_Fault_Handler
00000004 C i
         U main
000000f4 W MM_Fault_Handler
000000f4 W NMI_Handler
00000000 T Reset_Handler
         U S_bss
         U S_data
         U Stack_Top
000000f4 W Usage_Fault_Handler
00000000 D Vectors
```

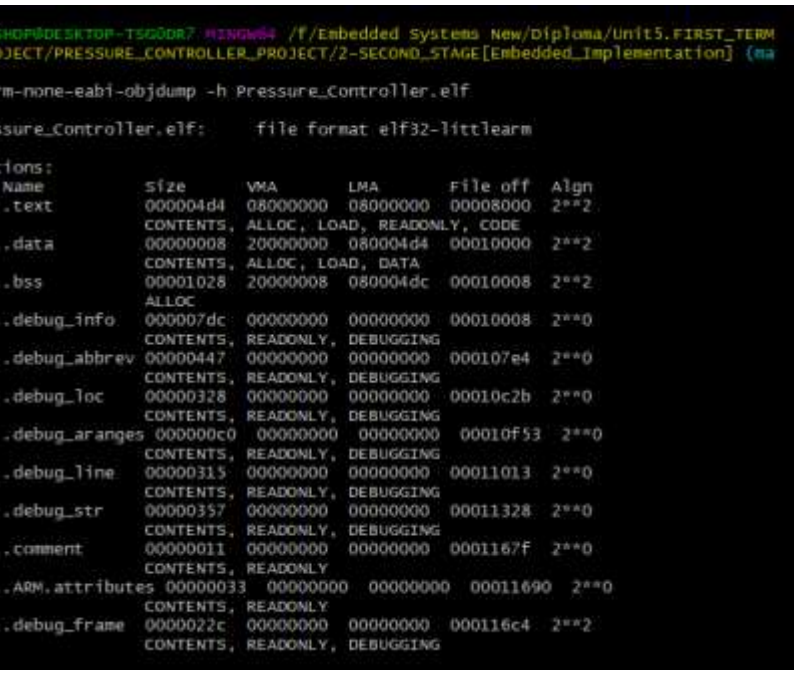
```
PC-SHOP@DESKTOP-TSGODR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-nm Pressure_Controller.elf
08000064 T AL_alarmSet
0800004c T AL_init
20001014 B AL_State
20001010 B AL_State_ID
20000000 D Alarm_Period
20001028 B Alarm_Val
20000008 B Alarm_Value
080004c8 W Bus_Fault_Handler
080004c8 T Default_Handler
0800009c T Delay
20000010 B E_bss
20000008 D E_data
080004d4 T E_text
080000c0 T getPressureVal
08000128 T GPIO_INITIALIZATION
080004c8 W Hard_Fault_Handler
2000102c B i
080002ec T MA_alarmOff
08000330 T MA_alarmOn
08000264 T MA_pressureDetection
20001018 B MA_State
20001020 B MA_State_ID
08000200 T main
080004c8 W MM_Fault_Handler
080004c8 W NMI_Handler
2000000c B P_Val
20001024 B P_Value
20000004 D Pressure_Threshold
08000374 T PS_init
0800038c T PS_read
2000101c B PS_State
20001021 B PS_State_ID
080003d4 T Reset_Handler
20000008 B S_bss
20000000 D S_data
080000d8 T Set_Alarm_actuator
0800001c T Set_Alarm_State
08000290 T Set_PressureVal
080001c0 T setup_app
20001010 B Stack_Top
080004c8 W Usage_Fault_Handler
08000000 T Vectors
```



15-Section_Tables:

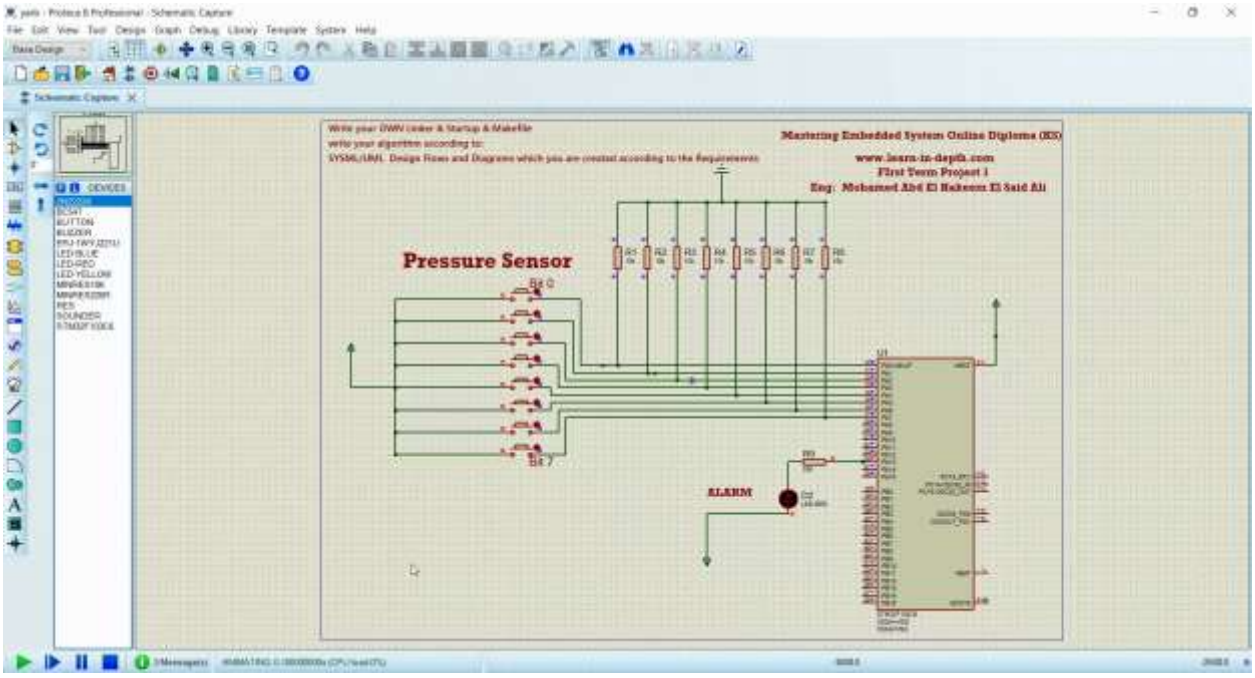
```
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-objdump -h Pressure_Sensor.o

Pressure_Sensor.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          00000004  00000000  00000000  00000094  2**2
                  ALLOC
  3 .debug_info   00000110  00000000  00000000  00000094  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000a5  00000000  00000000  000001a4  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000058  00000000  00000000  00000249  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  000002a1  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000005e  00000000  00000000  000002c1  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000177  00000000  00000000  0000031f  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  00000496  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000004a8  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000044  00000000  00000000  000004dc  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING

PC-SHOP@DESKTOP-TSGODR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-objdump -h Main_Algorithm.o

Main_Algorithm.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000110  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000008  00000000  00000000  00000144  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000014c  2**0
                  ALLOC
  3 .debug_info   00000176  00000000  00000000  0000014c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000bb  00000000  00000000  000002c2  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000000bc  00000000  00000000  0000037d  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000439  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000073  00000000  00000000  00000459  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000001e1  00000000  00000000  000004cc  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  000006ad  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000006bf  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000007c  00000000  00000000  000006f4  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

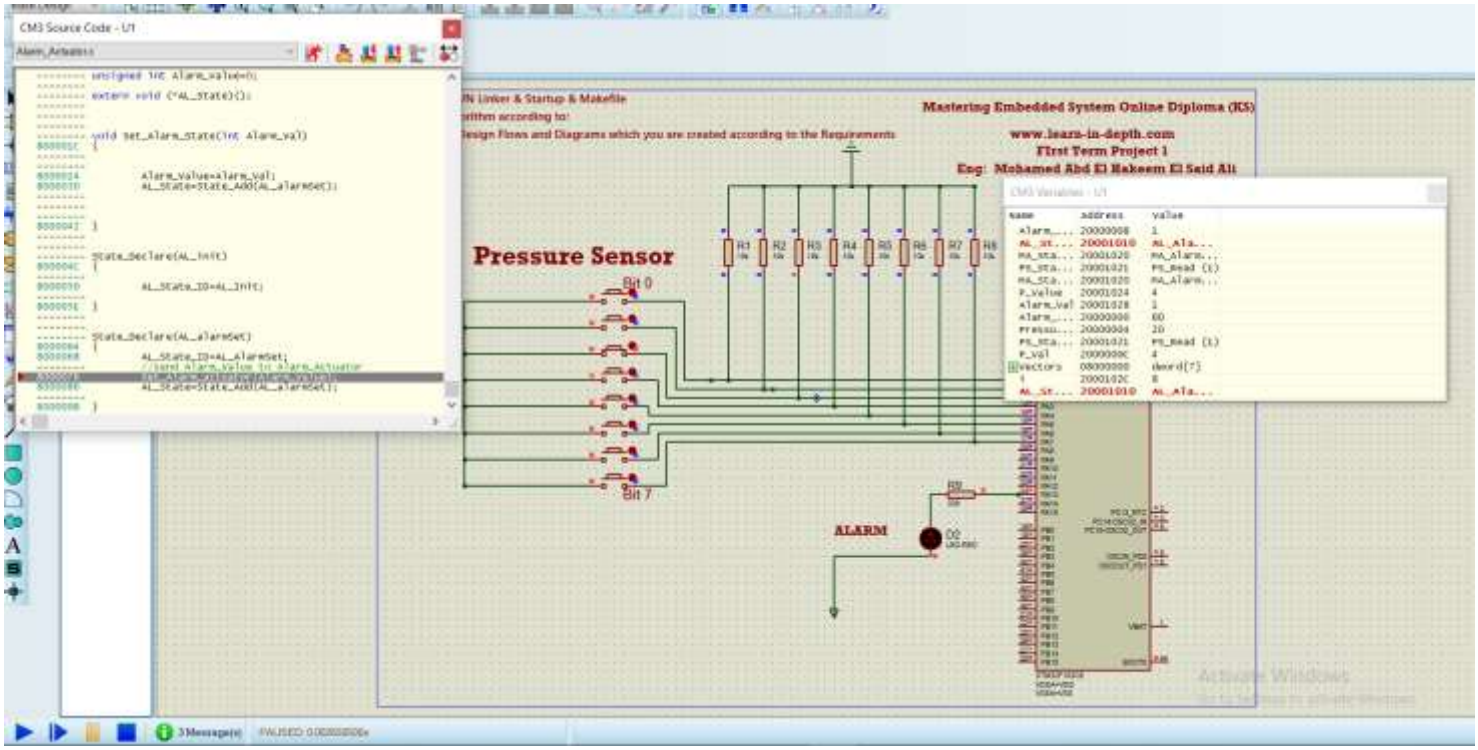
```
$ arm-none-eabi-objdump -h Alarm_Actuator.o

Alarm_Actuator.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000080  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000b4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  000000b4  2**2
                  ALLOC
  3 .debug_info   00000139  00000000  00000000  000000b4  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000ce  00000000  00000000  000001ed  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000090  00000000  00000000  000002bb  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  0000034b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000005f  00000000  00000000  0000036b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000001a2  00000000  00000000  000003ca  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  0000056c  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  0000057e  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000060  00000000  00000000  000005b4  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING

PC-SHOP@DESKTOP-TSGODR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-objdump -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000a4  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000d8  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000d8  2**0
                  ALLOC
  3 .debug_info   00000185  00000000  00000000  000000d8  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000b6  00000000  00000000  0000025d  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000058  00000000  00000000  00000313  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  0000036b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000089  00000000  00000000  0000038b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000001d0  00000000  00000000  00000414  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  000005e4  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000005f6  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000048  00000000  00000000  0000062c  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

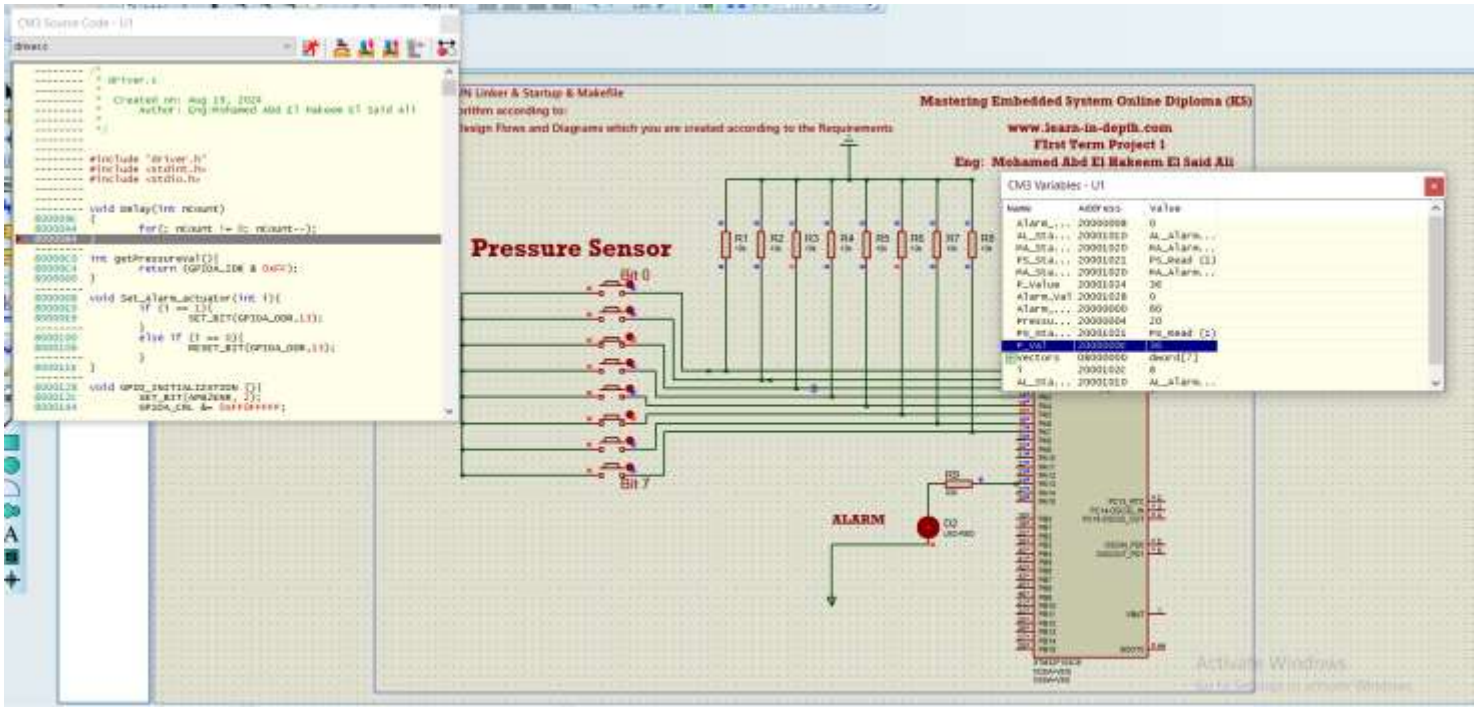


```
PC-SHOP@DESKTOP-TSGQDR7 MINGW64 /f/Embedded Systems New/Diploma/Unit5.FIRST_TERM
_PROJECT/PRESSURE_CONTROLLER_PROJECT/2-SECOND_STAGE[Embedded_Implementation] (ma
in)
$ arm-none-eabi-objdump -h Pressure_Controller.elf

Pressure_Controller.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000004d4  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  20000000  080004d4  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001028  20000008  080004dc  00010008  2**2
                  ALLOC
  3 .debug_info   000007dc  00000000  00000000  00010008  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 00000447  00000000  00000000  000107e4  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000328  00000000  00000000  00010c2b  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000c0  00000000  00000000  00010f53  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000315  00000000  00000000  00011013  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    00000357  00000000  00000000  00011328  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  0001167f  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00011690  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000022c  00000000  00000000  000116c4  2**2
                  CONTENTS, READONLY, DEBUGGING
```

3 Third Stage [Simulation in Proteus]:






Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
First Term Project 1
Eng: Mohamed Abd El Hakeem El Said Ali
Pressure Sensor
Bit 0
Bit 7
ALARM
D2
R1
R2
R3
R4
R5
R6
R7
R8
R9
CM3 Variables - U1
Name
Address
Value
Activate Windows