PRESSURE DETECTOR

Project 1

GitHub: Youssef-Adel-22/Master-Embedded-Systems/

MAY 31, 2023

YOUSSEF ADEL MOHAMED
Learn-in-depth

1. Case Study:

➢ System Description

Pressure controller informs crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin.

Alarm duration 60 seconds

Keeps track of the measured values (optional)

➢ Assumptions

- Controller set up and shutdown are not modeled.
- Controller maintenance not modeled.
- pressure sensor will not fail.
- alarm never fails.
- Controller never faces power cut

➢ Versioning

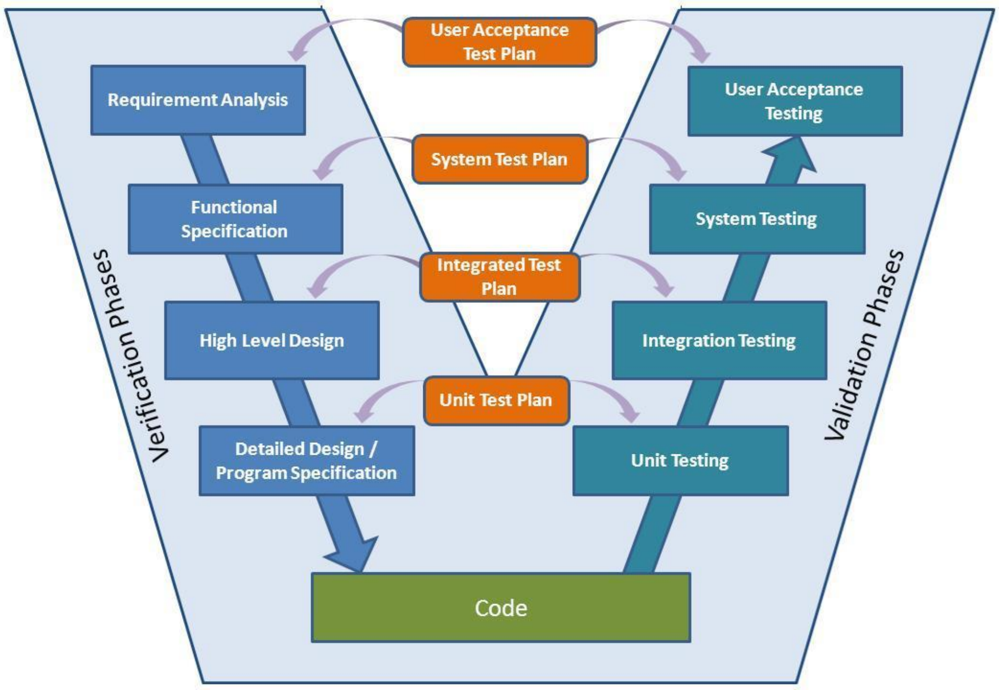
Keep track of measured values not modeled in first version.

➢ SOC

STM32F103C6

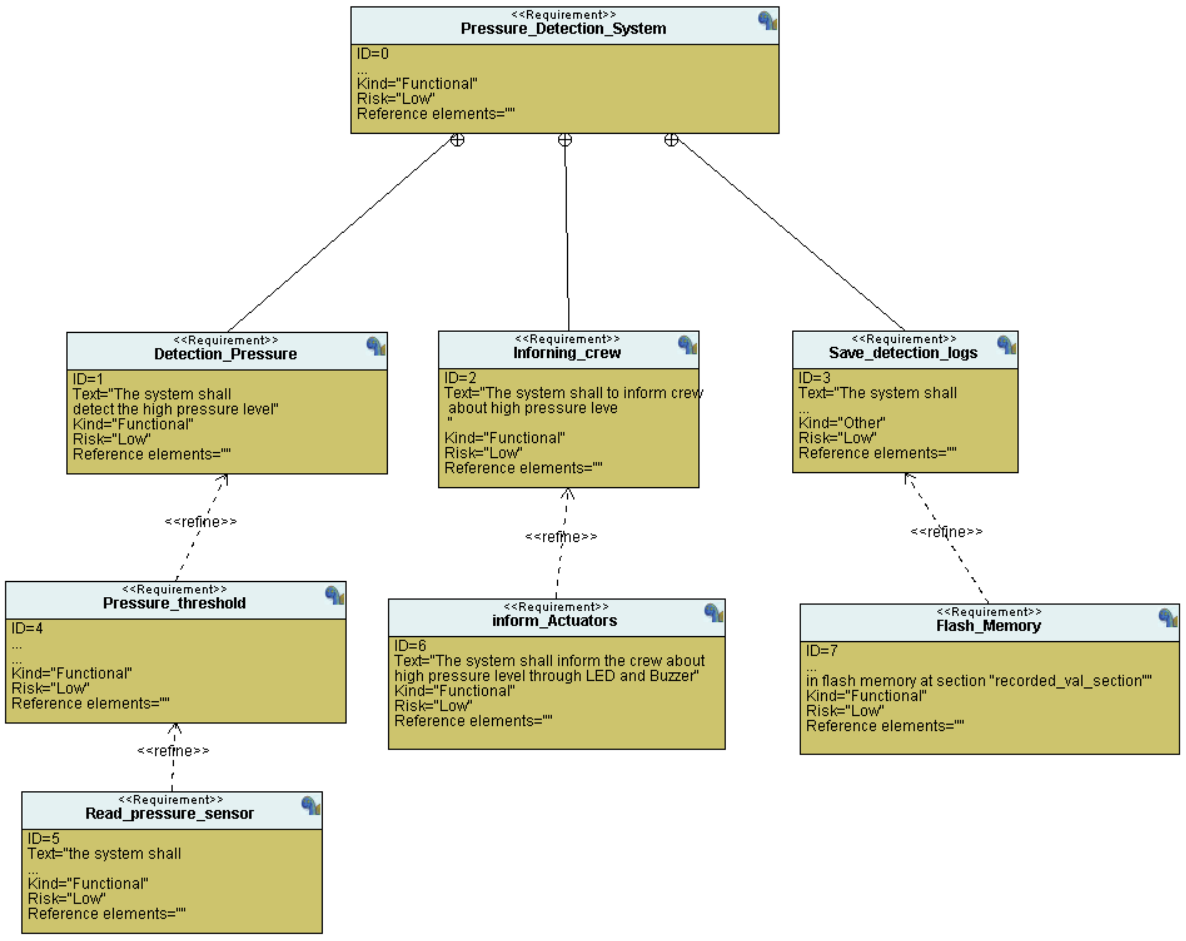
2. Method:

V-model



3. System Requirements:

Requirements Graph:

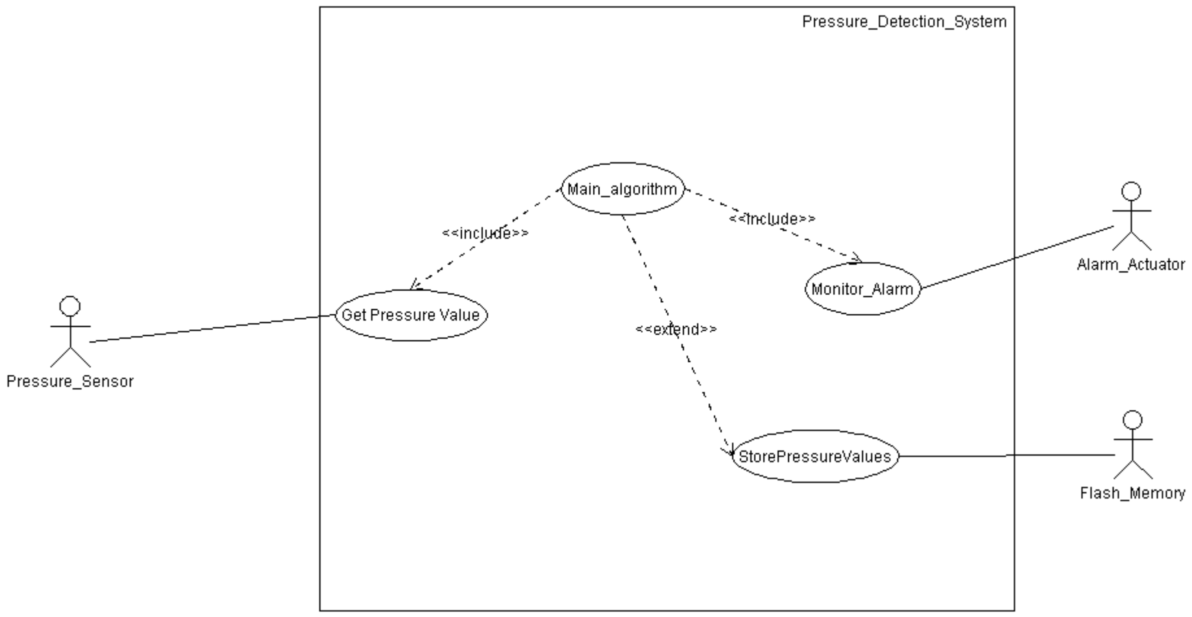


4. Space Exploration/Partitioning:

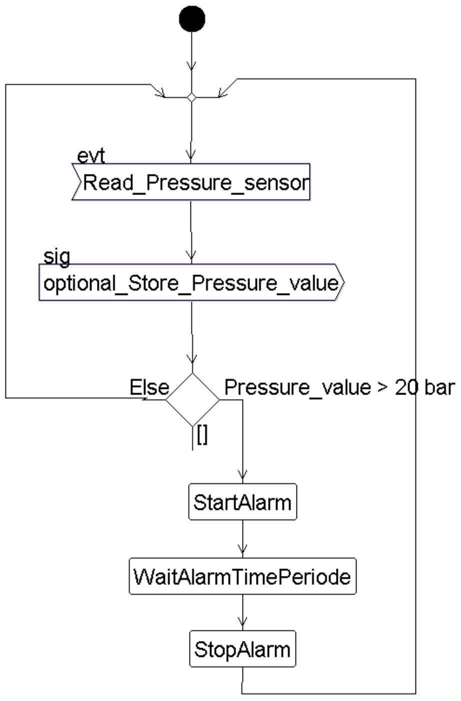
A single SOC Stm32 microcontroller with a cortex m3 processor will be used to implement this project.

System Analysis:

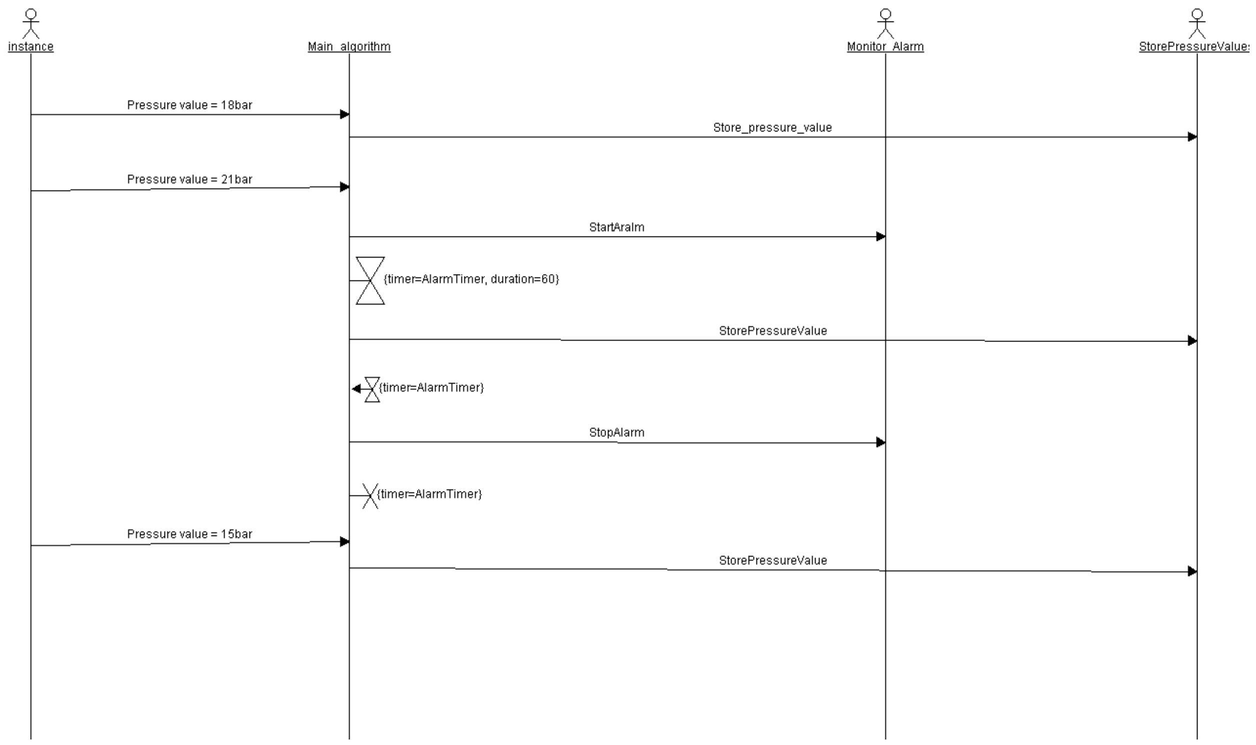
1. Use case diagram



2. Activity diagram

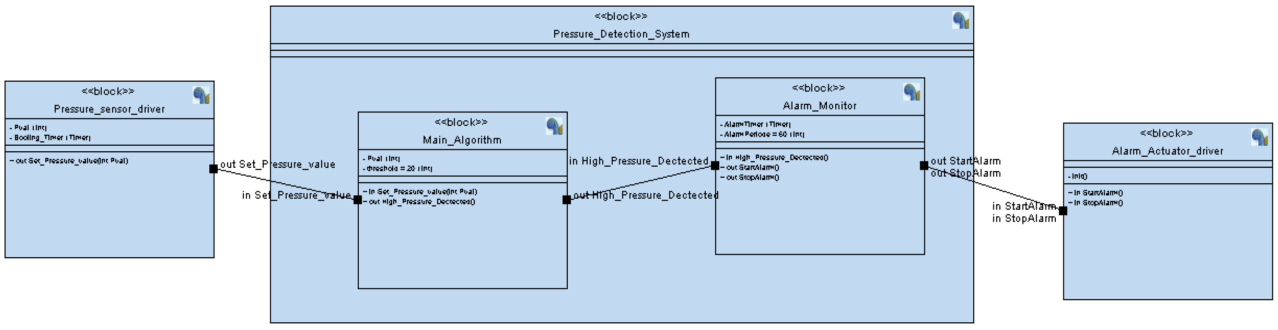


3. Sequence diagram



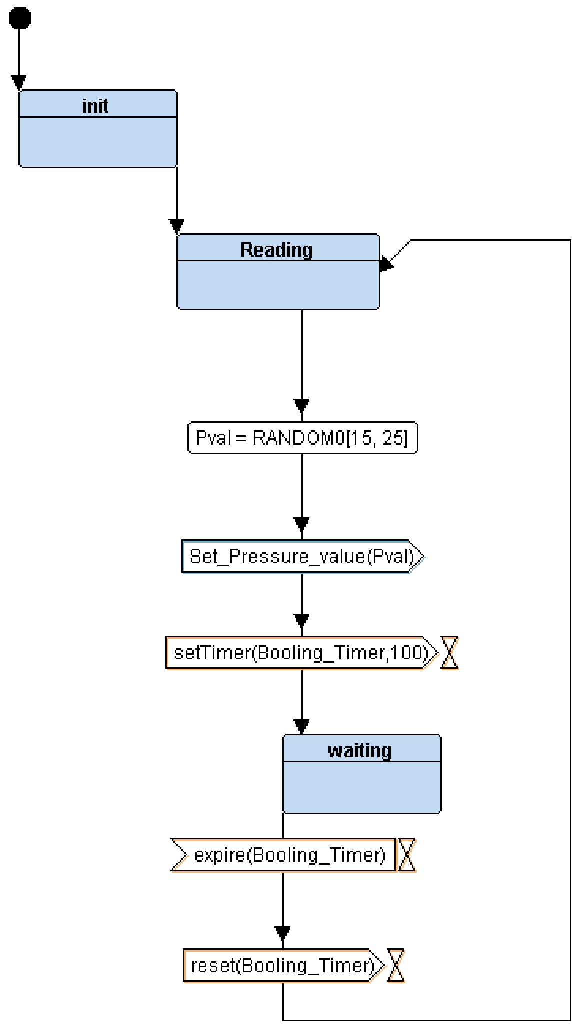
5. System design:

Block Diagram:

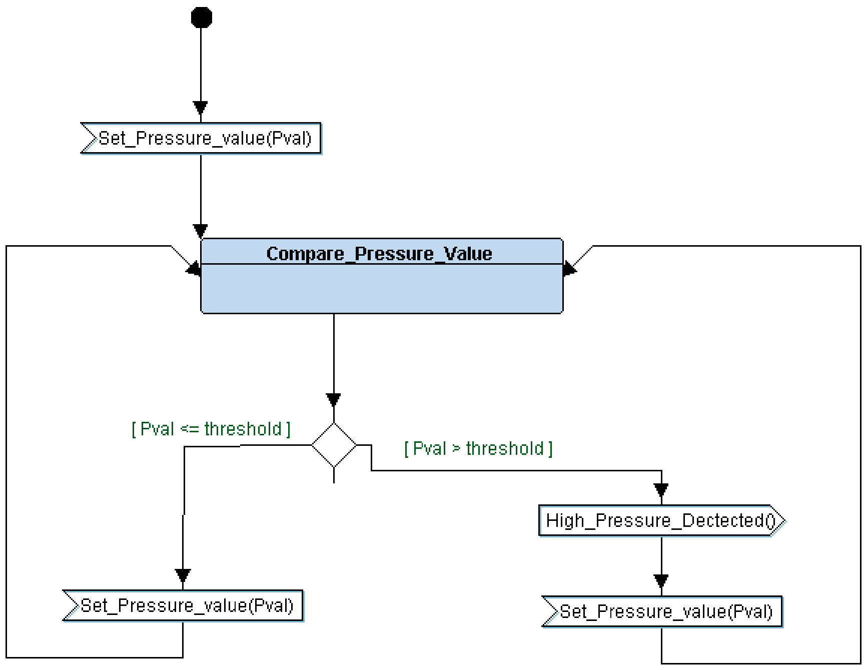


State machine:

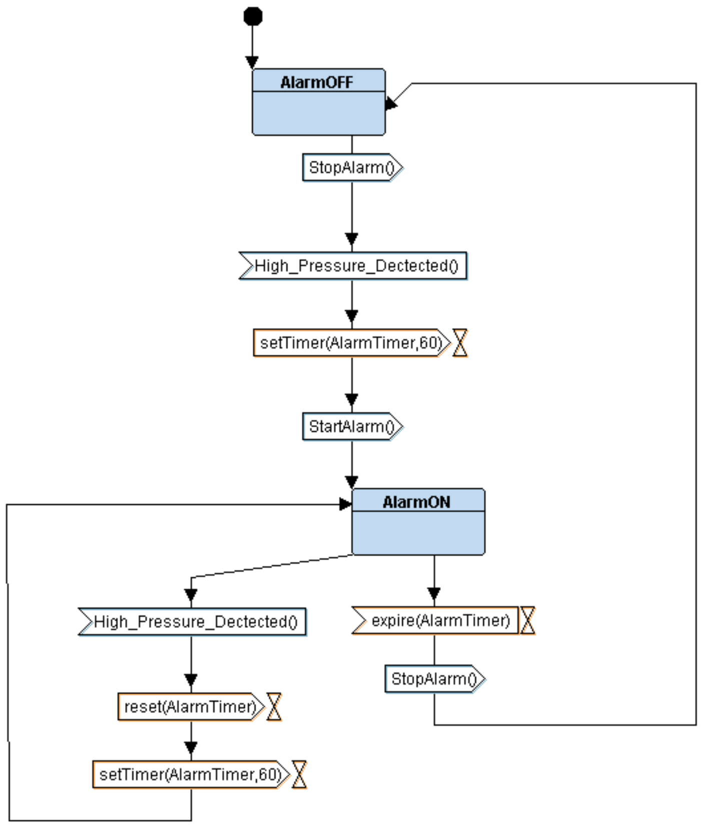
1. Pressure Sensor



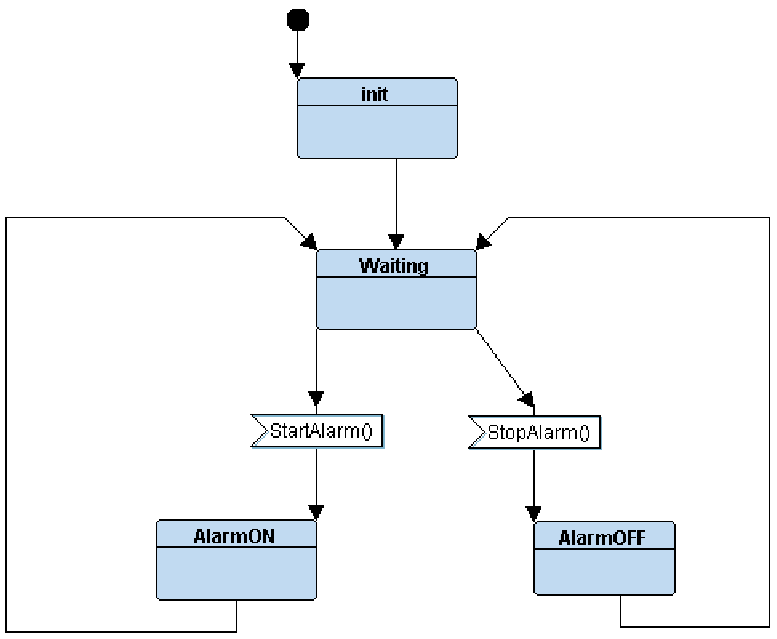
2. Main Algorithm



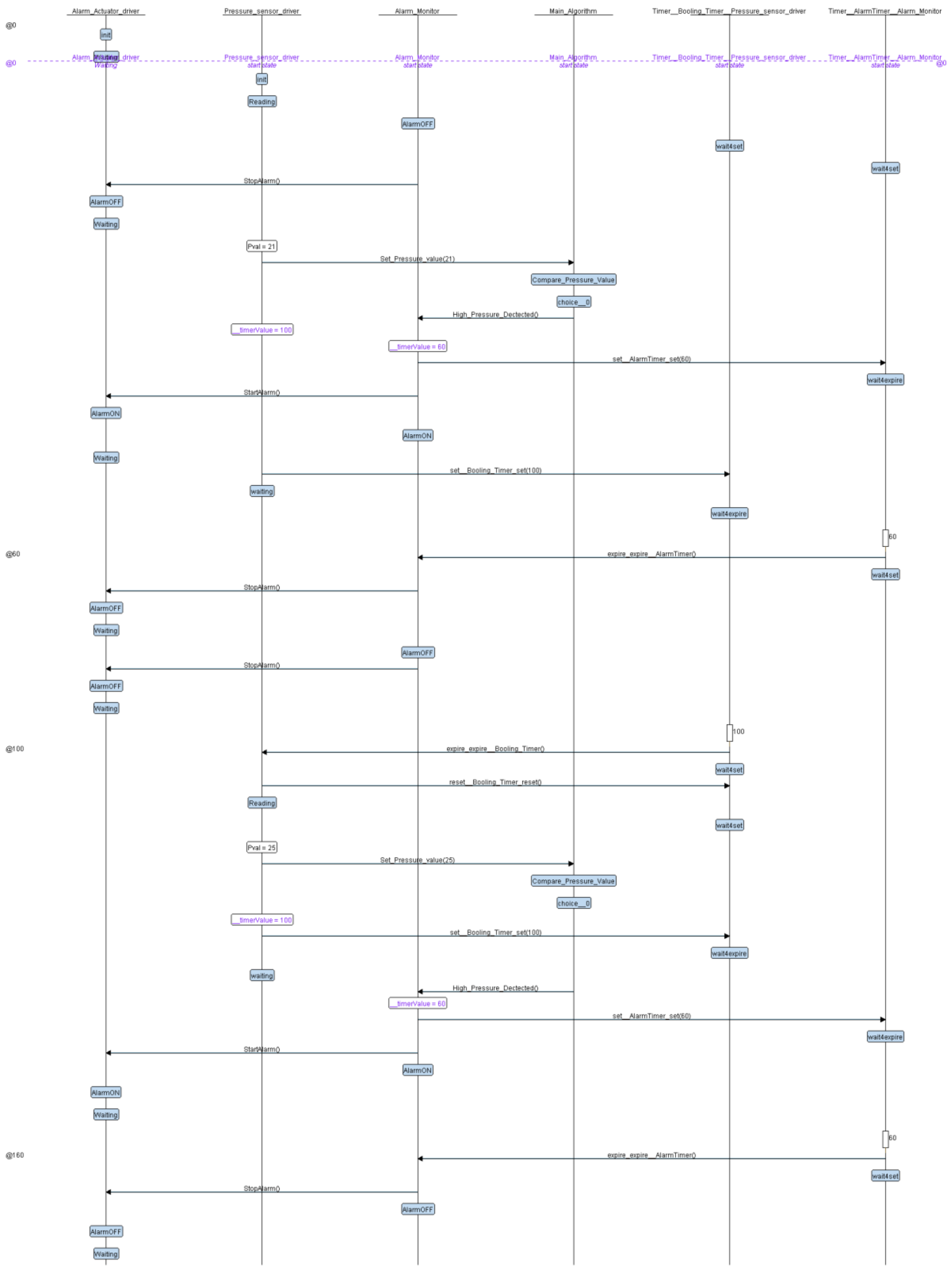
3. Alarm Monitor



4. Alarm Actuator

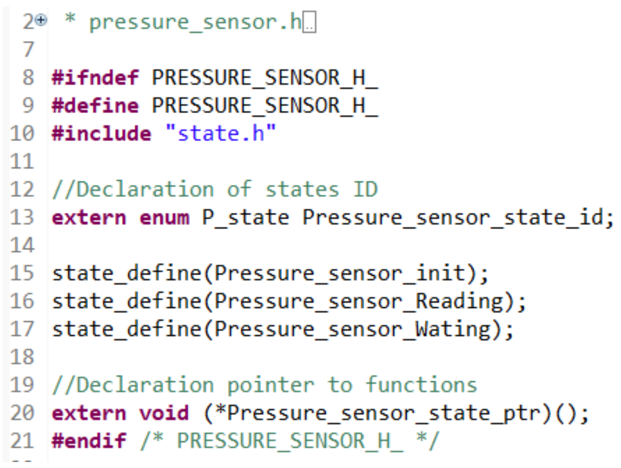


6. Simulation (UML) (V&V):

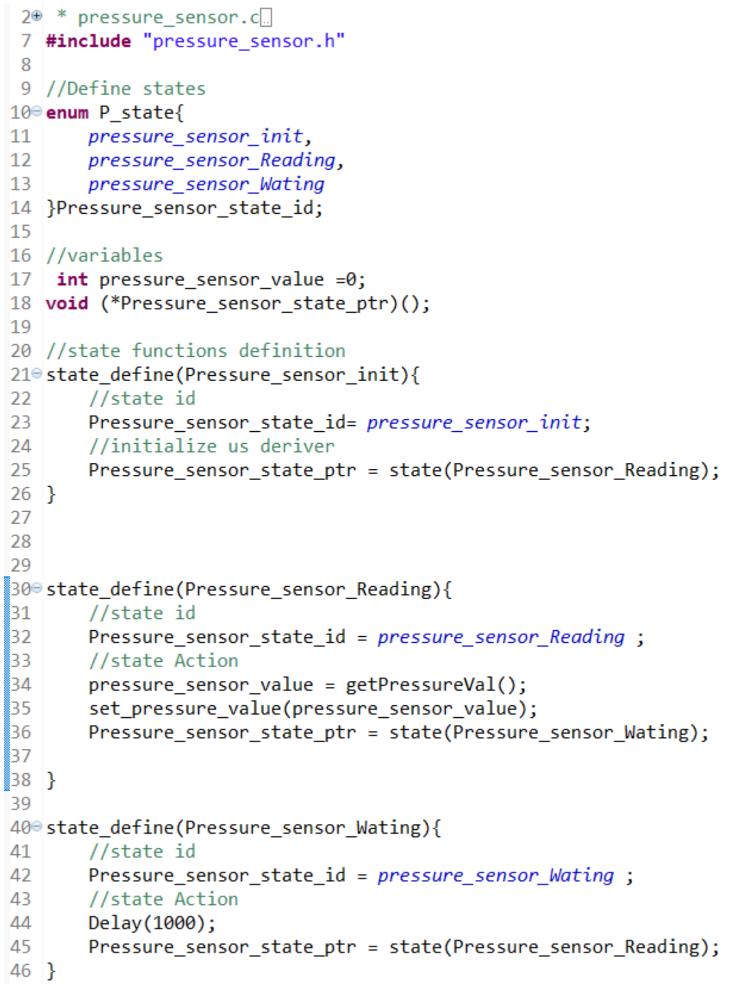


7. Codes, Startup, Linker script:

1) Pressure Sensor state

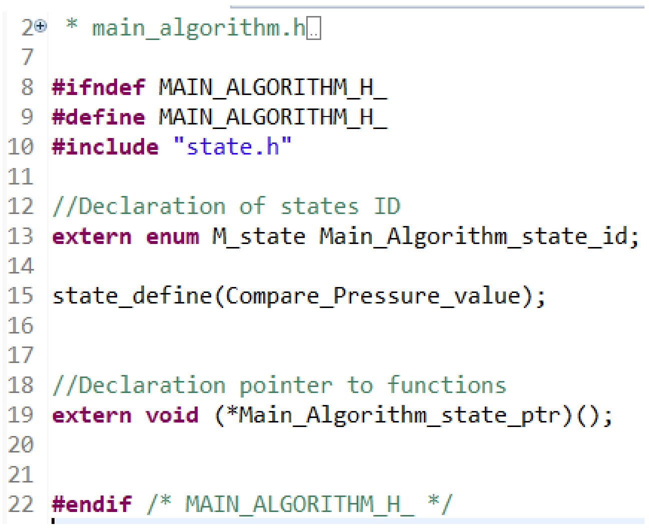


```c
 * pressure_sensor.h

#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_
#include "state.h"

//Declaration of states ID
extern enum P_state Pressure_sensor_state_id;

state_define(Pressure_sensor_init);
state_define(Pressure_sensor_Reading);
state_define(Pressure_sensor_Wating);

//Declaration pointer to functions
extern void (*Pressure_sensor_state_ptr)();
#endif /* PRESSURE_SENSOR_H_ */
```

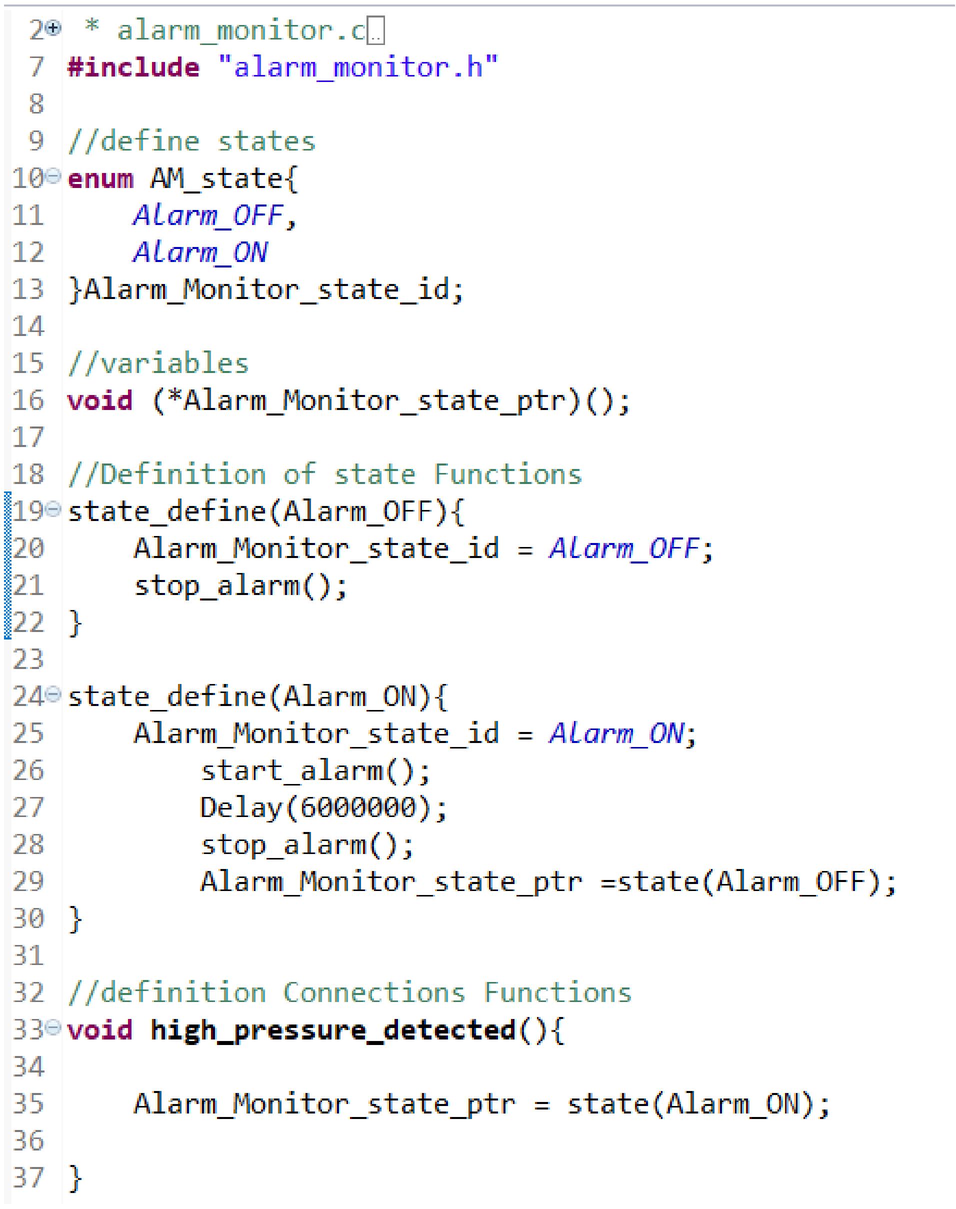


```c
 * pressure_sensor.c
#include "pressure_sensor.h"

//Define states
enum P_state{
    pressure_sensor_init,
    pressure_sensor_Reading,
    pressure_sensor_Wating
}Pressure_sensor_state_id;

//variables
 int pressure_sensor_value =0;
void (*Pressure_sensor_state_ptr)();

//state functions definition
state_define(Pressure_sensor_init){
    //state id
    Pressure_sensor_state_id= pressure_sensor_init;
    //initialize us deriver
    Pressure_sensor_state_ptr = state(Pressure_sensor_Reading);
}



state_define(Pressure_sensor_Reading){
    //state id
    Pressure_sensor_state_id = pressure_sensor_Reading ;
    //state Action
    pressure_sensor_value = getPressureVal();
    set_pressure_value(pressure_sensor_value);
    Pressure_sensor_state_ptr = state(Pressure_sensor_Wating);

}

state_define(Pressure_sensor_Wating){
    //state id
    Pressure_sensor_state_id = pressure_sensor_Wating ;
    //state Action
    Delay(1000);
    Pressure_sensor_state_ptr = state(Pressure_sensor_Reading);
}
```

2) Main Algorithm state code

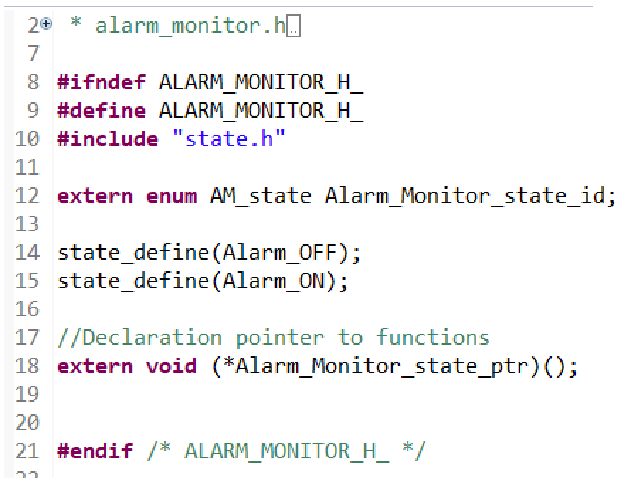


```
 * main_algorithm.h

#ifndef MAIN_ALGORITHM_H_
#define MAIN_ALGORITHM_H_
#include "state.h"

//Declaration of states ID
extern enum M_state Main_Algorithm_state_id;

state_define(Compare_Pressure_value);


//Declaration pointer to functions
extern void (*Main_Algorithm_state_ptr)();


#endif /* MAIN_ALGORITHM_H_ */
```

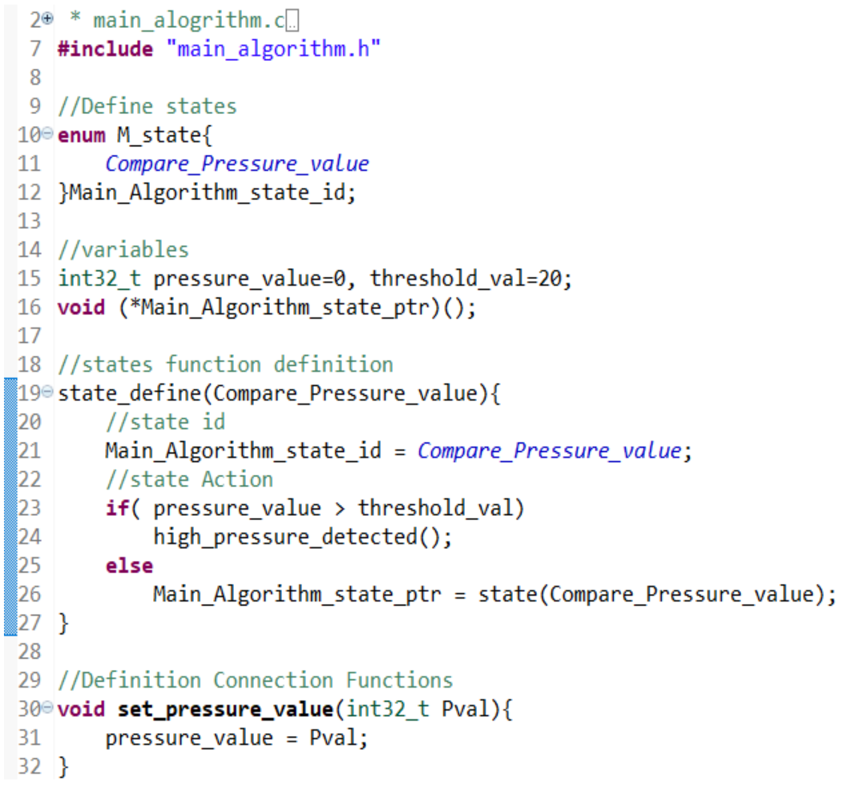


```
 * alarm_monitor.c
#include "alarm_monitor.h"

//define states
enum AM_state{
    Alarm_OFF,
    Alarm_ON
}Alarm_Monitor_state_id;

//variables
void (*Alarm_Monitor_state_ptr)();

//Definition of state Functions
state_define(Alarm_OFF){
    Alarm_Monitor_state_id = Alarm_OFF;
    stop_alarm();
}

state_define(Alarm_ON){
    Alarm_Monitor_state_id = Alarm_ON;
        start_alarm();
        Delay(6000000);
        stop_alarm();
        Alarm_Monitor_state_ptr =state(Alarm_OFF);
}

//definition Connections Functions
void high_pressure_detected(){

    Alarm_Monitor_state_ptr = state(Alarm_ON);

}
```

3) Alarm Monitor state code

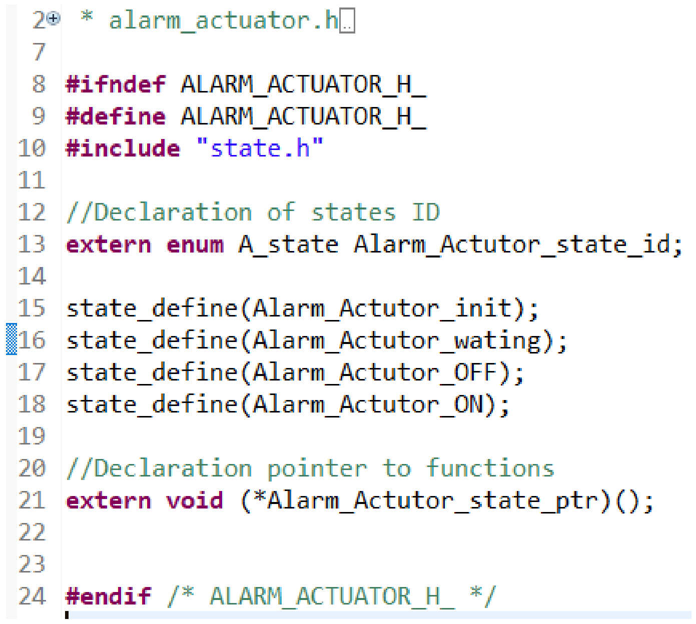


```c
 * alarm_monitor.h

#ifndef ALARM_MONITOR_H_
#define ALARM_MONITOR_H_
#include "state.h"

extern enum AM_state Alarm_Monitor_state_id;

state_define(Alarm_OFF);
state_define(Alarm_ON);

//Declaration pointer to functions
extern void (*Alarm_Monitor_state_ptr)();


#endif /* ALARM_MONITOR_H_ */
```

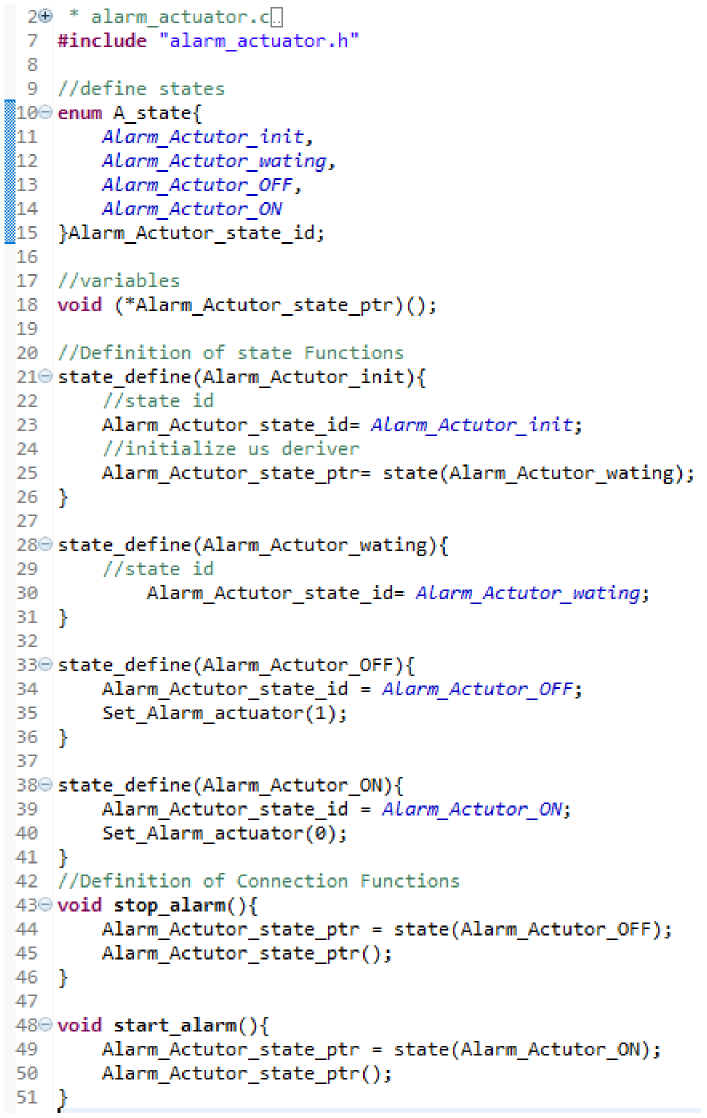


```c
 * main_alogrithm.c
#include "main_algorithm.h"

//Define states
enum M_state{
    Compare_Pressure_value
}Main_Algorithm_state_id;

//variables
int32_t pressure_value=0, threshold_val=20;
void (*Main_Algorithm_state_ptr)();

//states function definition
state_define(Compare_Pressure_value){
    //state id
    Main_Algorithm_state_id = Compare_Pressure_value;
    //state Action
    if( pressure_value > threshold_val)
        high_pressure_detected();
    else
        Main_Algorithm_state_ptr = state(Compare_Pressure_value);
}

//Definition Connection Functions
void set_pressure_value(int32_t Pval){
    pressure_value = Pval;
}
```

4) Alarm Acuator state code

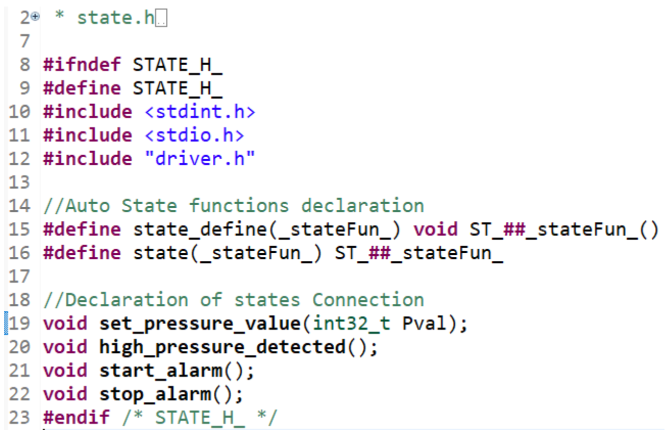


```c
 * alarm_actuator.h

#ifndef ALARM_ACTUATOR_H_
#define ALARM_ACTUATOR_H_
#include "state.h"

//Declaration of states ID
extern enum A_state Alarm_Actutor_state_id;

state_define(Alarm_Actutor_init);
state_define(Alarm_Actutor_wating);
state_define(Alarm_Actutor_OFF);
state_define(Alarm_Actutor_ON);

//Declaration pointer to functions
extern void (*Alarm_Actutor_state_ptr)();


#endif /* ALARM_ACTUATOR_H_ */
```

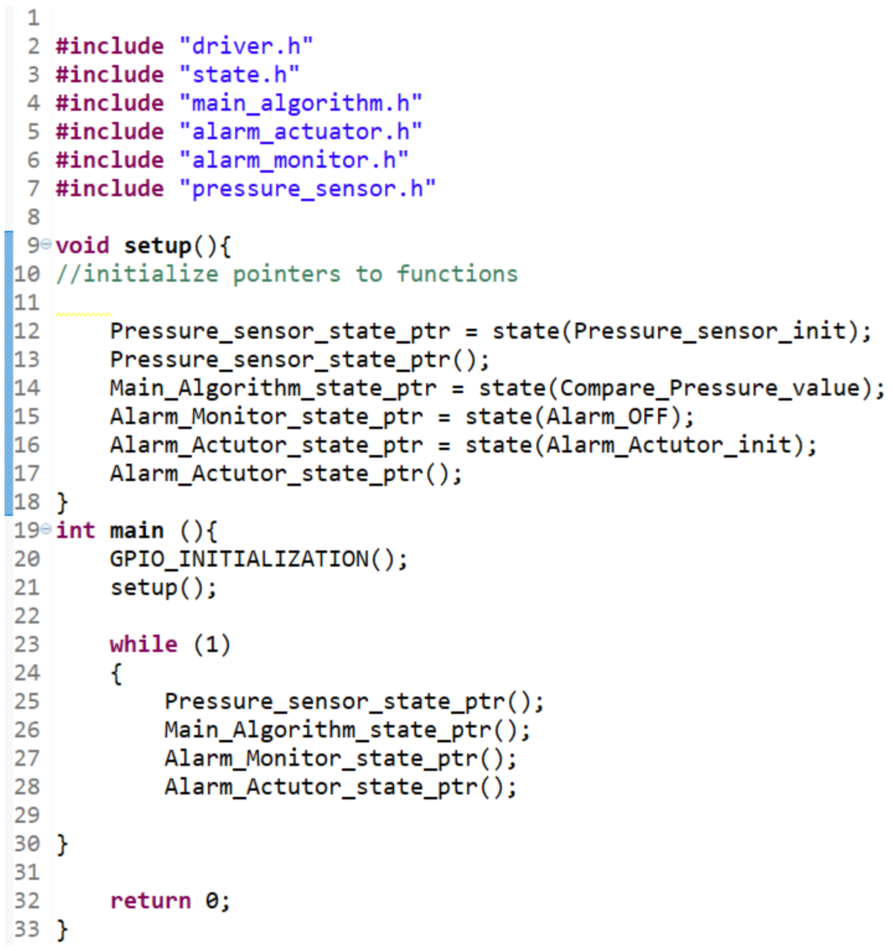


```c
 * alarm_actuator.c
#include "alarm_actuator.h"

//define states
enum A_state{
    Alarm_Actutor_init,
    Alarm_Actutor_wating,
    Alarm_Actutor_OFF,
    Alarm_Actutor_ON
}Alarm_Actutor_state_id;

//variables
void (*Alarm_Actutor_state_ptr)();

//Definition of state Functions
state_define(Alarm_Actutor_init){
    //state id
    Alarm_Actutor_state_id= Alarm_Actutor_init;
    //initialize us deriver
    Alarm_Actutor_state_ptr= state(Alarm_Actutor_wating);
}

state_define(Alarm_Actutor_wating){
    //state id
        Alarm_Actutor_state_id= Alarm_Actutor_wating;
}

state_define(Alarm_Actutor_OFF){
    Alarm_Actutor_state_id = Alarm_Actutor_OFF;
    Set_Alarm_actuator(1);
}

state_define(Alarm_Actutor_ON){
    Alarm_Actutor_state_id = Alarm_Actutor_ON;
    Set_Alarm_actuator(0);
}
//Definition of Connection Functions
void stop_alarm(){
    Alarm_Actutor_state_ptr = state(Alarm_Actutor_OFF);
    Alarm_Actutor_state_ptr();
}

void start_alarm(){
    Alarm_Actutor_state_ptr = state(Alarm_Actutor_ON);
    Alarm_Actutor_state_ptr();
}
```

5) State and main code

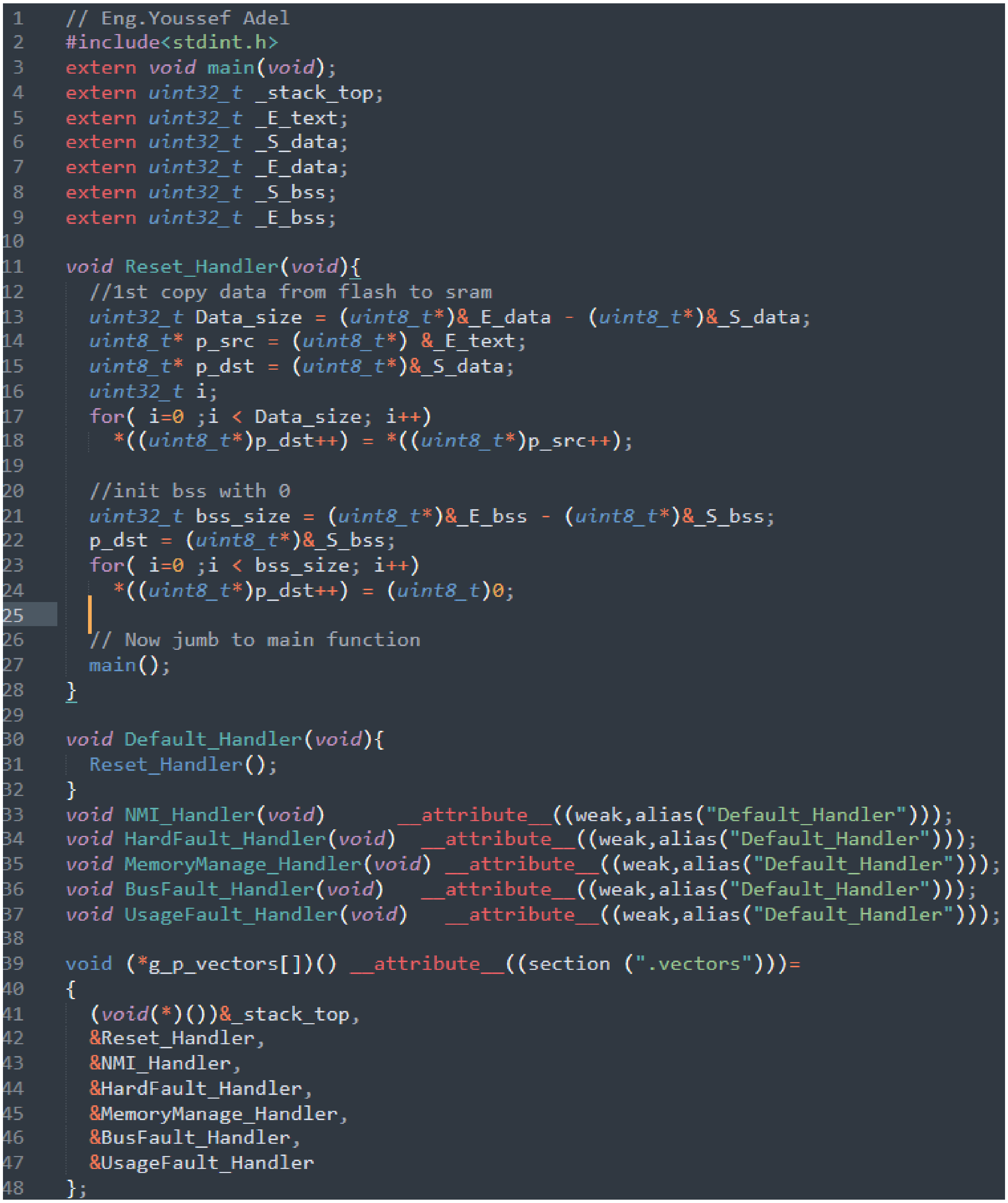


```c
 * state.h

#ifndef STATE_H_
#define STATE_H_
#include <stdint.h>
#include <stdio.h>
#include "driver.h"

//Auto State functions declaration
#define state_define(_stateFun_) void ST_##_stateFun_()
#define state(_stateFun_) ST_##_stateFun_

//Declaration of states Connection
void set_pressure_value(int32_t Pval);
void high_pressure_detected();
void start_alarm();
void stop_alarm();
#endif /* STATE_H_ */
```



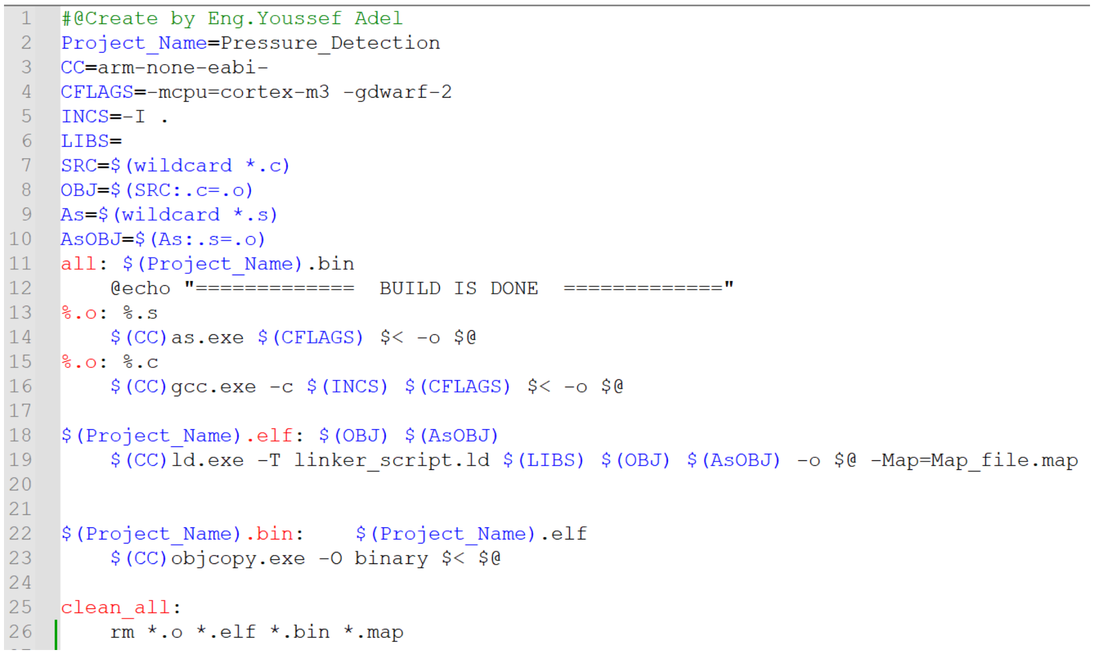
```c

#include "driver.h"
#include "state.h"
#include "main_algorithm.h"
#include "alarm_actuator.h"
#include "alarm_monitor.h"
#include "pressure_sensor.h"

void setup(){
//initialize pointers to functions

    Pressure_sensor_state_ptr = state(Pressure_sensor_init);
    Pressure_sensor_state_ptr();
    Main_Algorithm_state_ptr = state(Compare_Pressure_value);
    Alarm_Monitor_state_ptr = state(Alarm_OFF);
    Alarm_Actutor_state_ptr = state(Alarm_Actutor_init);
    Alarm_Actutor_state_ptr();
}
int main (){
    GPIO_INITIALIZATION();
    setup();

    while (1)
    {
        Pressure_sensor_state_ptr();
        Main_Algorithm_state_ptr();
        Alarm_Monitor_state_ptr();
        Alarm_Actutor_state_ptr();

}

    return 0;
}
```

6) Startup code

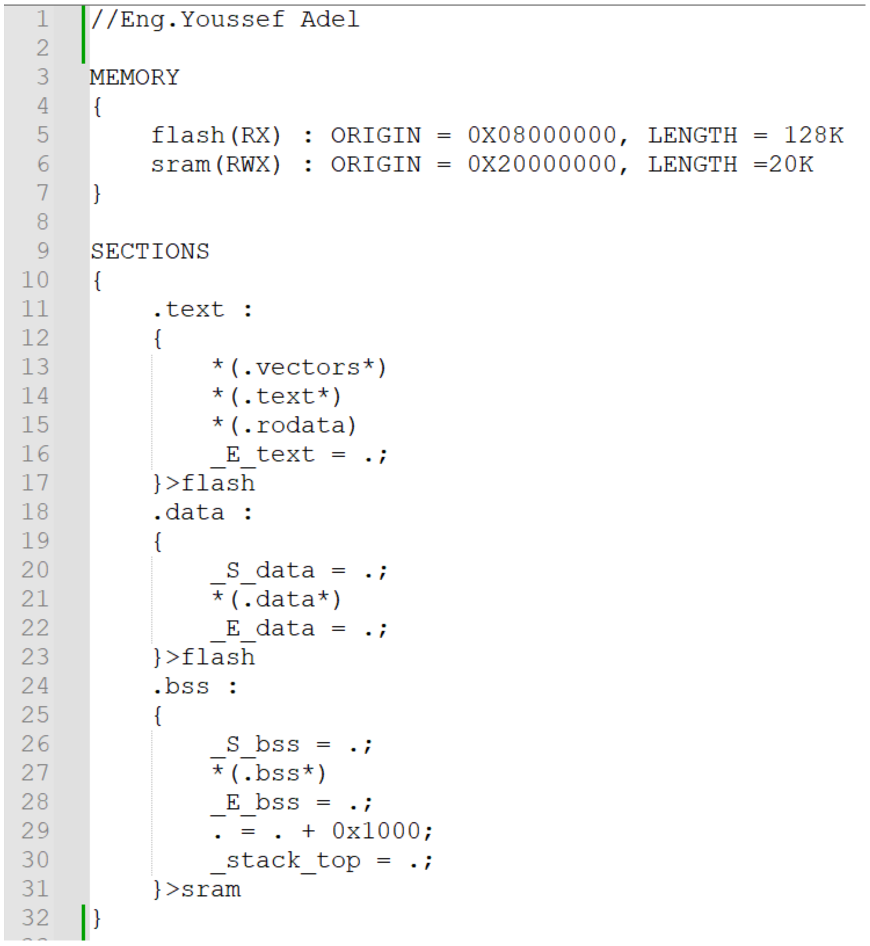


```c
// Eng.Youssef Adel
#include<stdint.h>
extern void main(void);
extern uint32_t _stack_top;
extern uint32_t _E_text;
extern uint32_t _S_data;
extern uint32_t _E_data;
extern uint32_t _S_bss;
extern uint32_t _E_bss;

void Reset_Handler(void){
  //1st copy data from flash to sram
  uint32_t Data_size = (uint8_t*)&_E_data - (uint8_t*)&_S_data;
  uint8_t* p_src = (uint8_t*) &_E_text;
  uint8_t* p_dst = (uint8_t*)&_S_data;
  uint32_t i;
  for( i=0 ;i < Data_size; i++)
    *((uint8_t*)p_dst++) = *((uint8_t*)p_src++);

  //init bss with 0
  uint32_t bss_size = (uint8_t*)&_E_bss - (uint8_t*)&_S_bss;
  p_dst = (uint8_t*)&_S_bss;
  for( i=0 ;i < bss_size; i++)
    *((uint8_t*)p_dst++) = (uint8_t)0;

  // Now jumb to main function
  main();
}

void Default_Handler(void){
  Reset_Handler();
}
void NMI_Handler(void)       __attribute__((weak,alias("Default_Handler")));
void HardFault_Handler(void)  __attribute__((weak,alias("Default_Handler")));
void MemoryManage_Handler(void) __attribute__((weak,alias("Default_Handler")));
void BusFault_Handler(void)   __attribute__((weak,alias("Default_Handler")));
void UsageFault_Handler(void)   __attribute__((weak,alias("Default_Handler")));

void (*g_p_vectors[])() __attribute__((section (".vectors")))=
{
  (void(*)())&_stack_top,
  &Reset_Handler,
  &NMI_Handler,
  &HardFault_Handler,
  &MemoryManage_Handler,
  &BusFault_Handler,
  &UsageFault_Handler
};
```

7) Make file and Linker



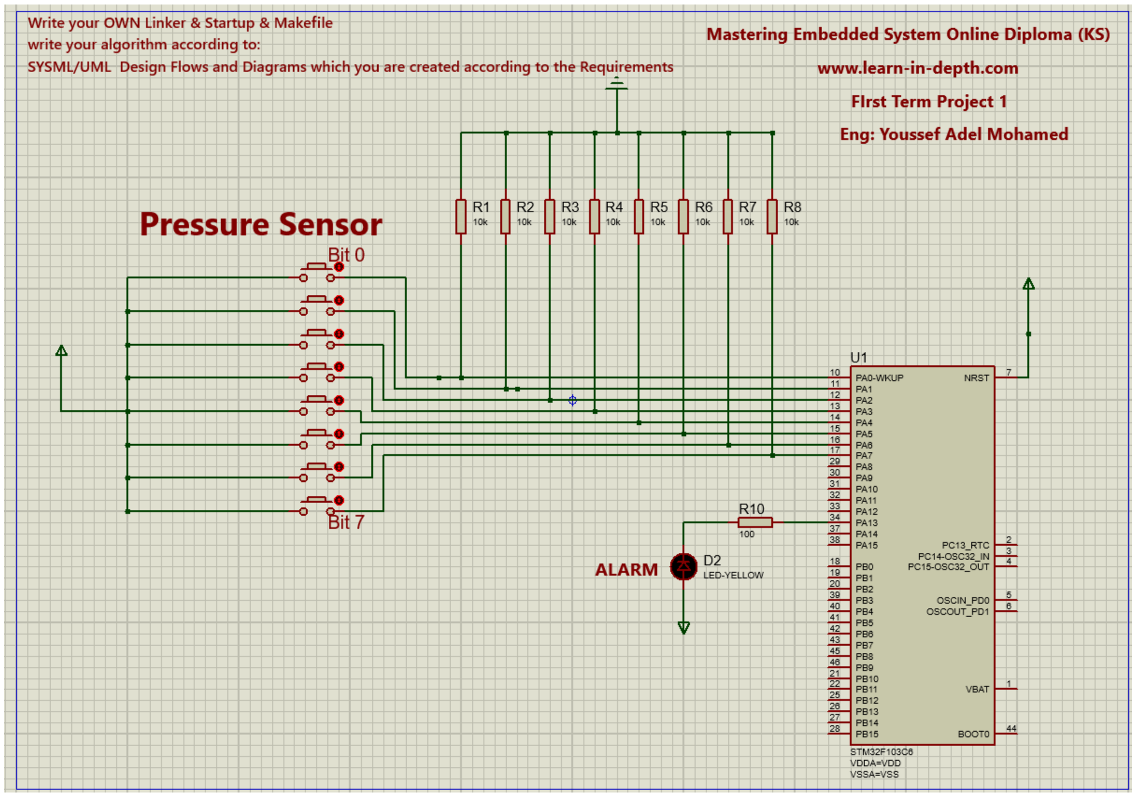
```makefile
#@Create by Eng.Youssef Adel
Project_Name=Pressure_Detection
CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -gdwarf-2
INCS=-I .
LIBS=
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
As=$(wildcard *.s)
AsOBJ=$(As:.s=.o)
all: $(Project_Name).bin
    @echo "=============  BUILD IS DONE  ============="
%.o: %.s
    $(CC)as.exe $(CFLAGS) $< -o $@
%.o: %.c
    $(CC)gcc.exe -c $(INCS) $(CFLAGS) $< -o $@

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


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

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



```
//Eng.Youssef Adel

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 = .;
    }>flash
    .bss :
    {
        _S_bss = .;
        *(.bss*)
        _E_bss = .;
        . = . + 0x1000;
        _stack_top = .;
    }>sram
}
```

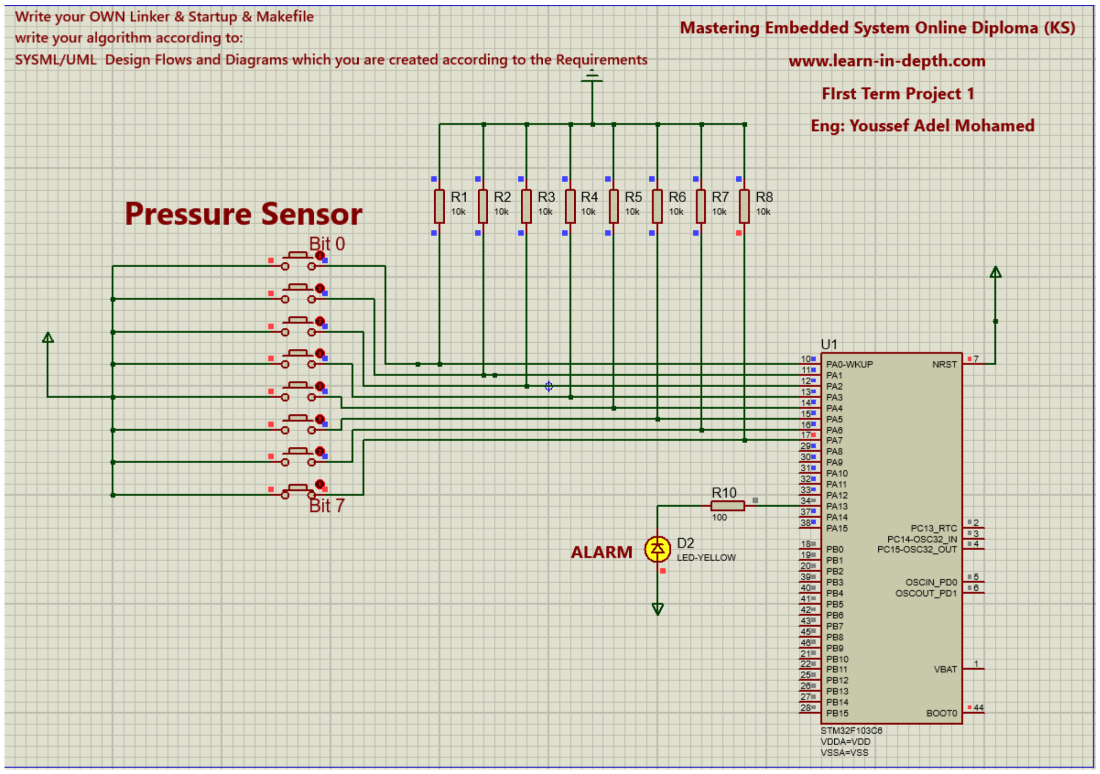
8. Proteus Simulation:

Pressure less than threshold:



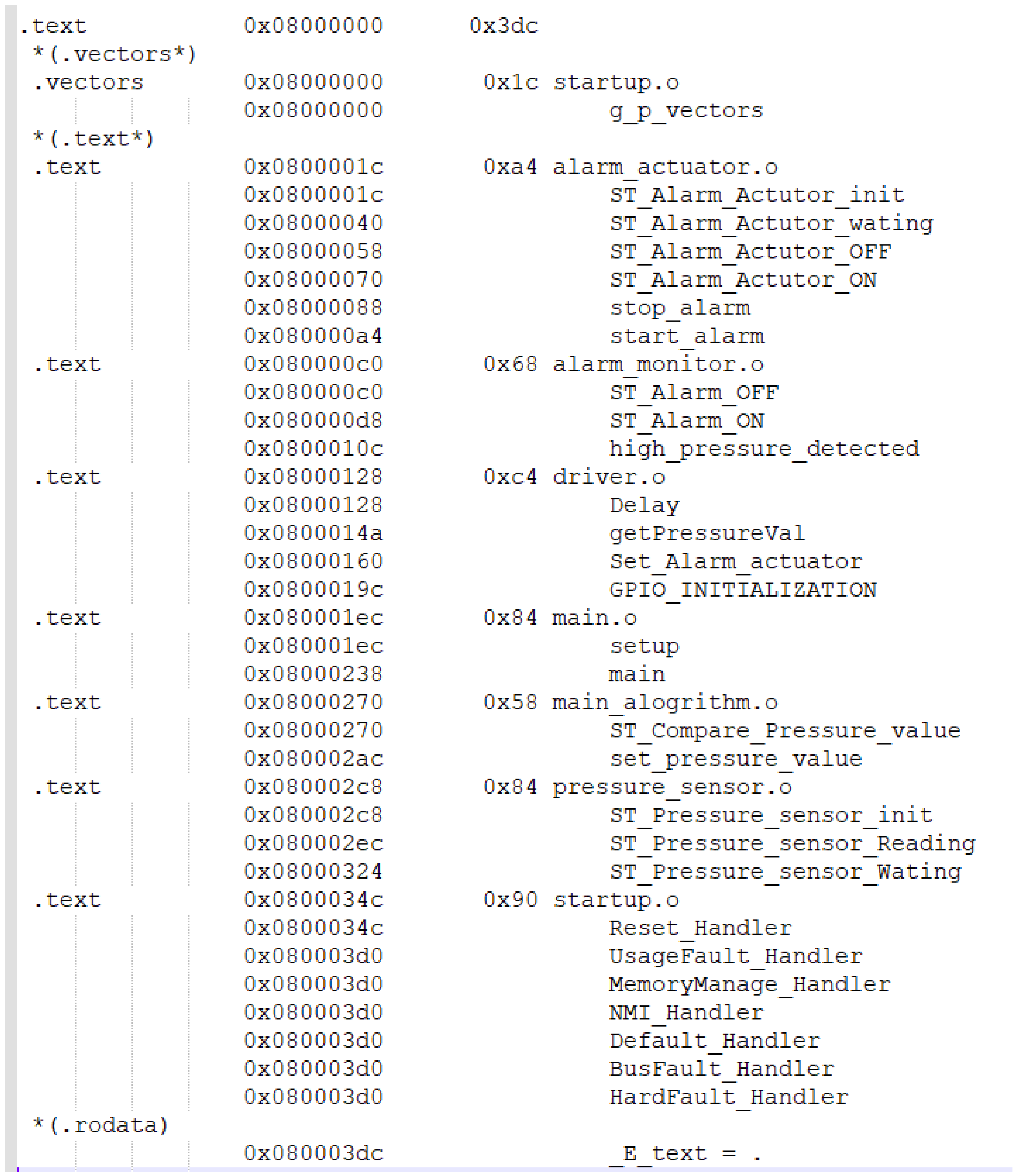
Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements
Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Eng: Youssef Adel Mohamed
Pressure Sensor
Bit 0
Bit 7
R1 10k
R2 10k
R3 10k
R4 10k
R5 10k
R6 10k
R7 10k
R8 10k
R10 100
ALARM
D2 LED-YELLOW
U1
STM32F103C6
VDDA=VDD
VSSA=VSS

Pressure more than threshold:

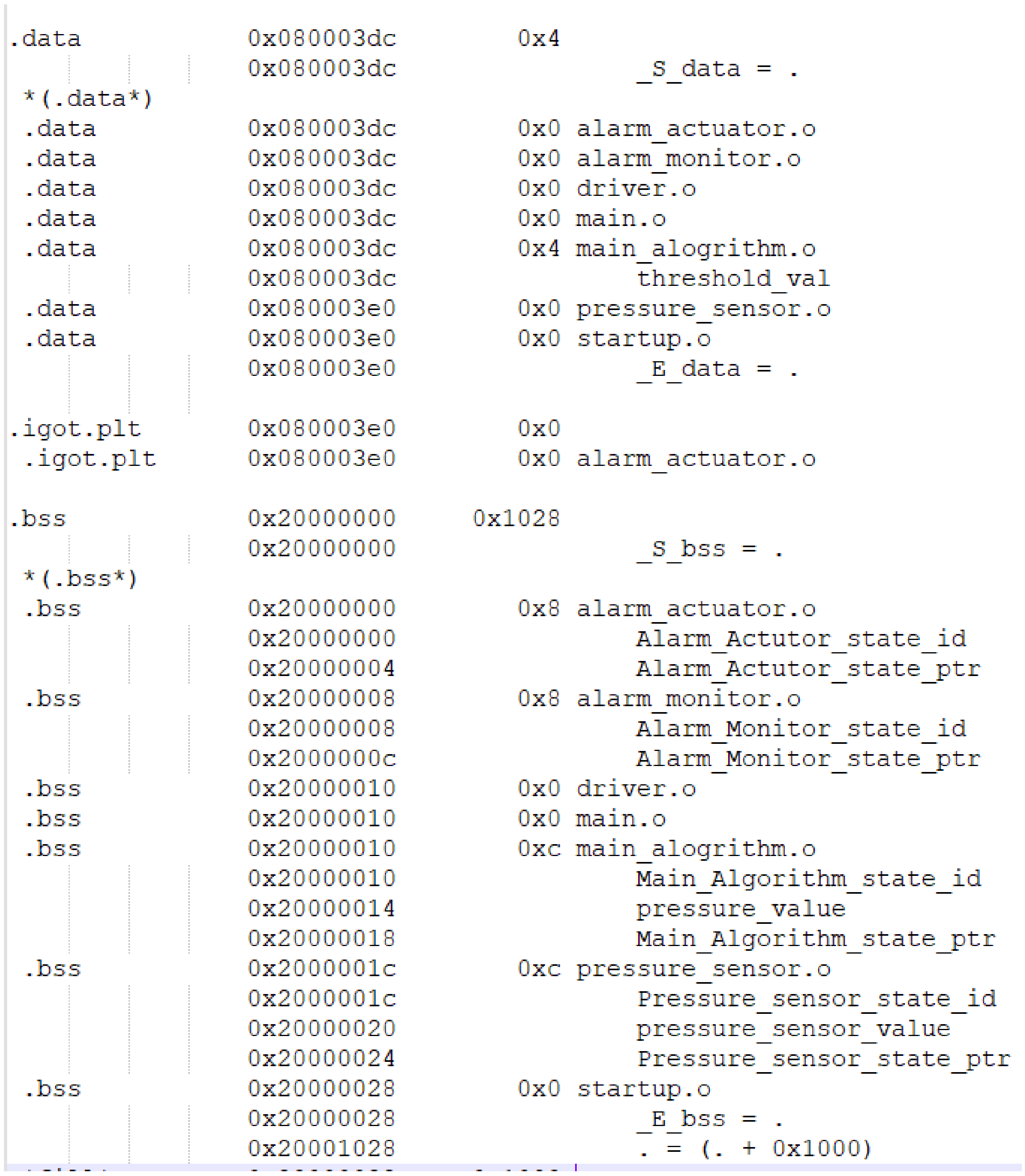


Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements
Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Eng: Youssef Adel Mohamed
Pressure Sensor
Bit 0
Bit 7
R1 10k
R2 10k
R3 10k
R4 10k
R5 10k
R6 10k
R7 10k
R8 10k
R10 100
ALARM
D2 LED-YELLOW
U1
STM32F103C6
VDDA=VDD
VSSA=VSS

9. Map file, symbol tables, Section tables:

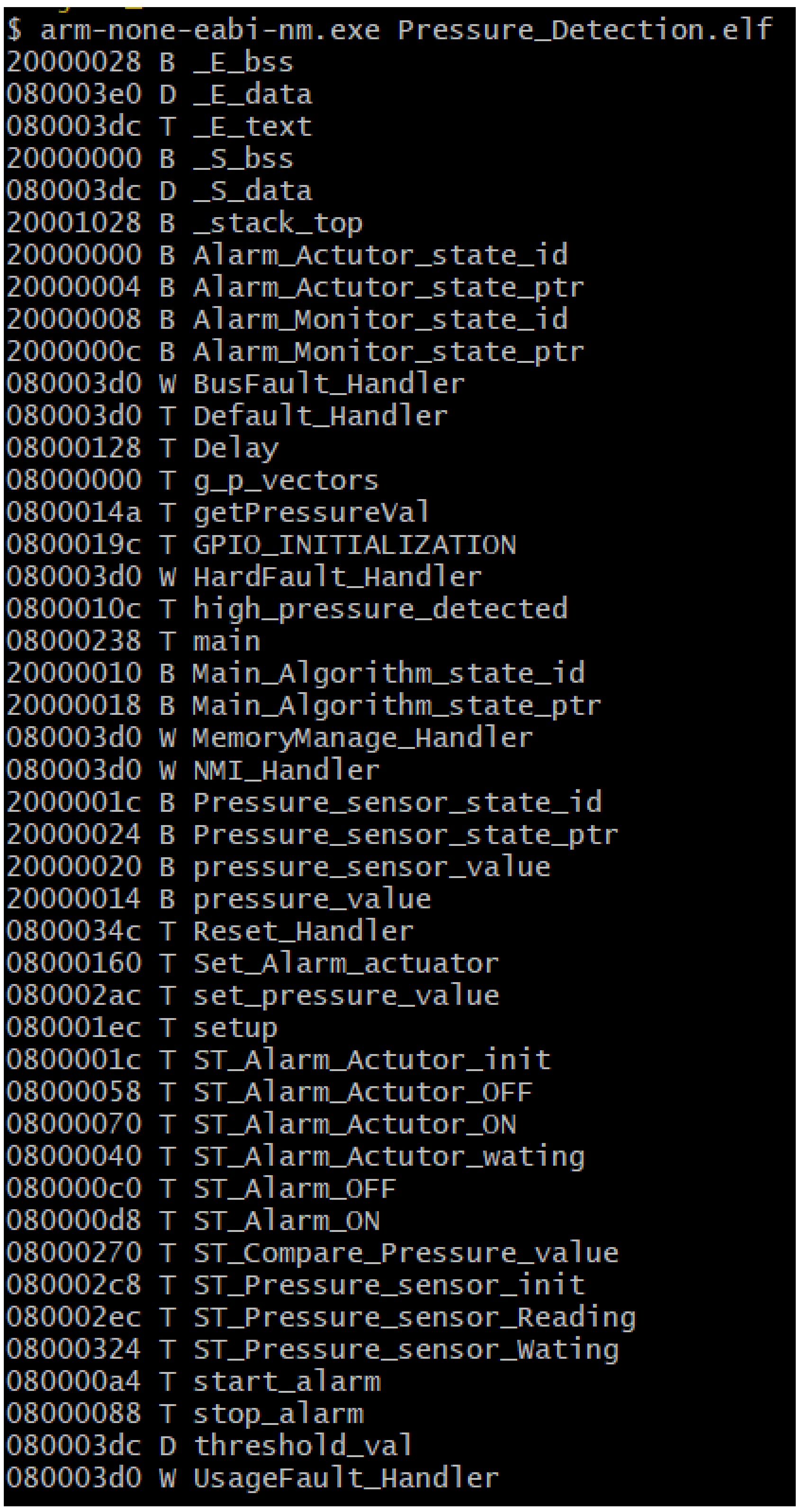


```
.text           0x08000000      0x3dc
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                g_p_vectors
 *(.text*)
 .text          0x0800001c       0xa4 alarm_actuator.o
                0x0800001c                ST_Alarm_Actutor_init
                0x08000040                ST_Alarm_Actutor_wating
                0x08000058                ST_Alarm_Actutor_OFF
                0x08000070                ST_Alarm_Actutor_ON
                0x08000088                stop_alarm
                0x080000a4                start_alarm
 .text          0x080000c0       0x68 alarm_monitor.o
                0x080000c0                ST_Alarm_OFF
                0x080000d8                ST_Alarm_ON
                0x0800010c                high_pressure_detected
 .text          0x08000128       0xc4 driver.o
                0x08000128                Delay
                0x0800014a                getPressureVal
                0x08000160                Set_Alarm_actuator
                0x0800019c                GPIO_INITIALIZATION
 .text          0x080001ec       0x84 main.o
                0x080001ec                setup
                0x08000238                main
 .text          0x08000270       0x58 main_alogrithm.o
                0x08000270                ST_Compare_Pressure_value
                0x080002ac                set_pressure_value
 .text          0x080002c8       0x84 pressure_sensor.o
                0x080002c8                ST_Pressure_sensor_init
                0x080002ec                ST_Pressure_sensor_Reading
                0x08000324                ST_Pressure_sensor_Wating
 .text          0x0800034c       0x90 startup.o
                0x0800034c                Reset_Handler
                0x080003d0                UsageFault_Handler
                0x080003d0                MemoryManage_Handler
                0x080003d0                NMI_Handler
                0x080003d0                Default_Handler
                0x080003d0                BusFault_Handler
                0x080003d0                HardFault_Handler
 *(.rodata)
                0x080003dc                _E_text = .
```



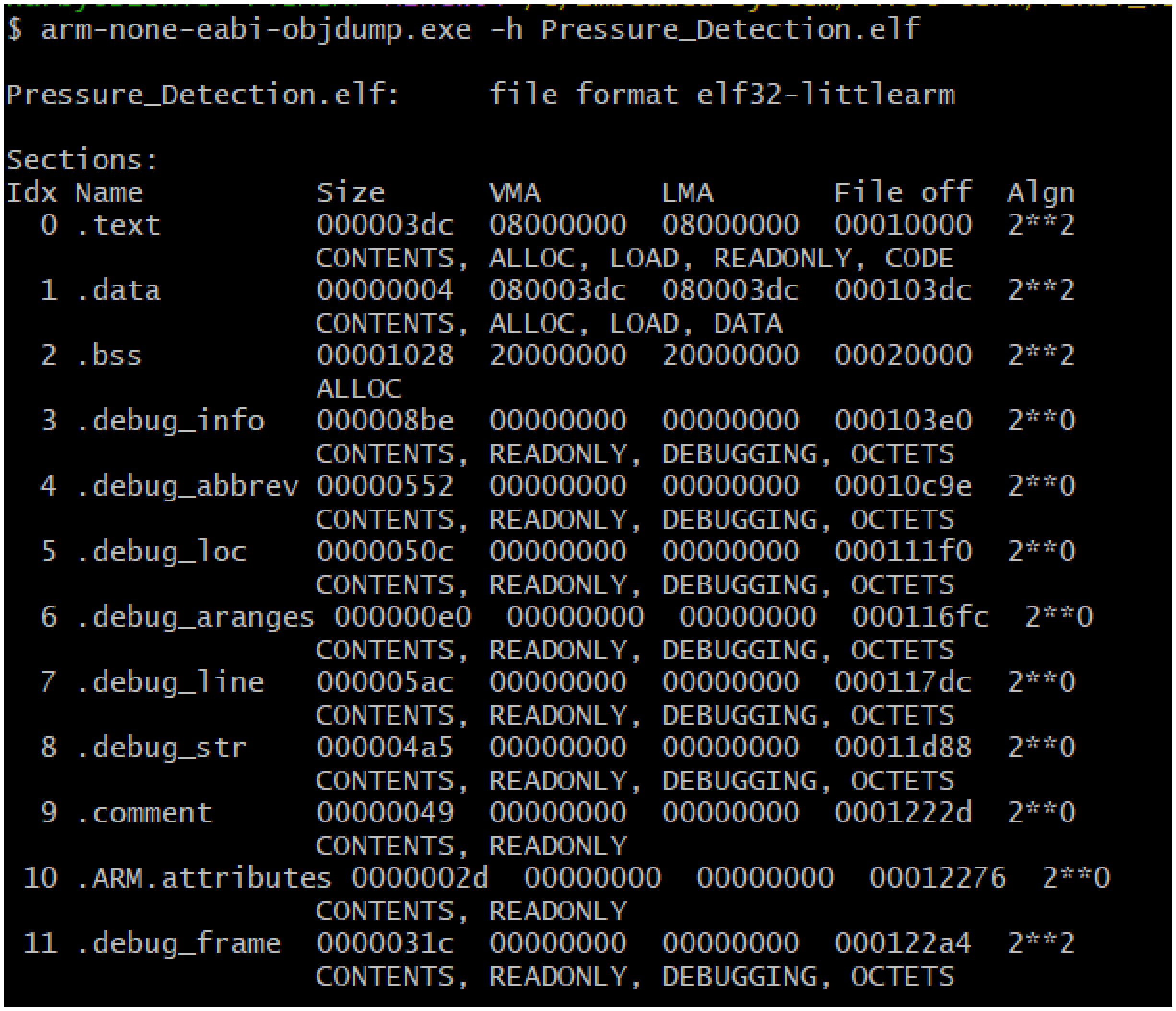
```
.data           0x080003dc        0x4
                0x080003dc                _S_data = .
 *(.data*)
 .data          0x080003dc        0x0 alarm_actuator.o
 .data          0x080003dc        0x0 alarm_monitor.o
 .data          0x080003dc        0x0 driver.o
 .data          0x080003dc        0x0 main.o
 .data          0x080003dc        0x4 main_alogrithm.o
                0x080003dc                threshold_val
 .data          0x080003e0        0x0 pressure_sensor.o
 .data          0x080003e0        0x0 startup.o
                0x080003e0                _E_data = .

.igot.plt       0x080003e0        0x0
 .igot.plt      0x080003e0        0x0 alarm_actuator.o

.bss            0x20000000     0x1028
                0x20000000                _S_bss = .
 *(.bss*)
 .bss           0x20000000        0x8 alarm_actuator.o
                0x20000000                Alarm_Actutor_state_id
                0x20000004                Alarm_Actutor_state_ptr
 .bss           0x20000008        0x8 alarm_monitor.o
                0x20000008                Alarm_Monitor_state_id
                0x2000000c                Alarm_Monitor_state_ptr
 .bss           0x20000010        0x0 driver.o
 .bss           0x20000010        0x0 main.o
 .bss           0x20000010        0xc main_alogrithm.o
                0x20000010                Main_Algorithm_state_id
                0x20000014                pressure_value
                0x20000018                Main_Algorithm_state_ptr
 .bss           0x2000001c        0xc pressure_sensor.o
                0x2000001c                Pressure_sensor_state_id
                0x20000020                pressure_sensor_value
                0x20000024                Pressure_sensor_state_ptr
 .bss           0x20000028        0x0 startup.o
                0x20000028                _E_bss = .
                0x20001028                . = (. + 0x1000)
```

Symbols of Pressure_Detection.elf



```
$ arm-none-eabi-nm.exe Pressure_Detection.elf
20000028 B _E_bss
080003e0 D _E_data
080003dc T _E_text
20000000 B _S_bss
080003dc D _S_data
20001028 B _stack_top
20000000 B Alarm_Actutor_state_id
20000004 B Alarm_Actutor_state_ptr
20000008 B Alarm_Monitor_state_id
2000000c B Alarm_Monitor_state_ptr
080003d0 W BusFault_Handler
080003d0 T Default_Handler
08000128 T Delay
08000000 T g_p_vectors
0800014a T getPressureVal
0800019c T GPIO_INITIALIZATION
080003d0 W HardFault_Handler
0800010c T high_pressure_detected
08000238 T main
20000010 B Main_Algorithm_state_id
20000018 B Main_Algorithm_state_ptr
080003d0 W MemoryManage_Handler
080003d0 W NMI_Handler
2000001c B Pressure_sensor_state_id
20000024 B Pressure_sensor_state_ptr
20000020 B pressure_sensor_value
20000014 B pressure_value
0800034c T Reset_Handler
08000160 T Set_Alarm_actuator
080002ac T set_pressure_value
080001ec T setup
0800001c T ST_Alarm_Actutor_init
08000058 T ST_Alarm_Actutor_OFF
08000070 T ST_Alarm_Actutor_ON
08000040 T ST_Alarm_Actutor_wating
080000c0 T ST_Alarm_OFF
080000d8 T ST_Alarm_ON
08000270 T ST_Compare_Pressure_value
080002c8 T ST_Pressure_sensor_init
080002ec T ST_Pressure_sensor_Reading
08000324 T ST_Pressure_sensor_Wating
080000a4 T start_alarm
08000088 T stop_alarm
080003dc D threshold_val
080003d0 W UsageFault_Handler
```

sections of 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         000003dc  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  080003dc  080003dc  000103dc  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001028  20000000  20000000  00020000  2**2
                  ALLOC
  3 .debug_info   000008be  00000000  00000000  000103e0  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  4 .debug_abbrev 00000552  00000000  00000000  00010c9e  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  5 .debug_loc    0000050c  00000000  00000000  000111f0  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  6 .debug_aranges 000000e0  00000000  00000000  000116fc  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  7 .debug_line   000005ac  00000000  00000000  000117dc  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  8 .debug_str    000004a5  00000000  00000000  00011d88  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  9 .comment      00000049  00000000  00000000  0001222d  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 0000002d  00000000  00000000  00012276  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000031c  00000000  00000000  000122a4  2**2
                  CONTENTS, READONLY, DEBUGGING, OCTETS
```