Mastering Embedded System Online Diploma

www.learn-in-depth.com

Topic	**First term (Final Project1)** **High Pressure Detector Report**
Name	**Ahmed Azazy Mohamed**
My Profile	https://www.learn-in-depth.com/online-diploma/ahmedazazyez%40gmail.com

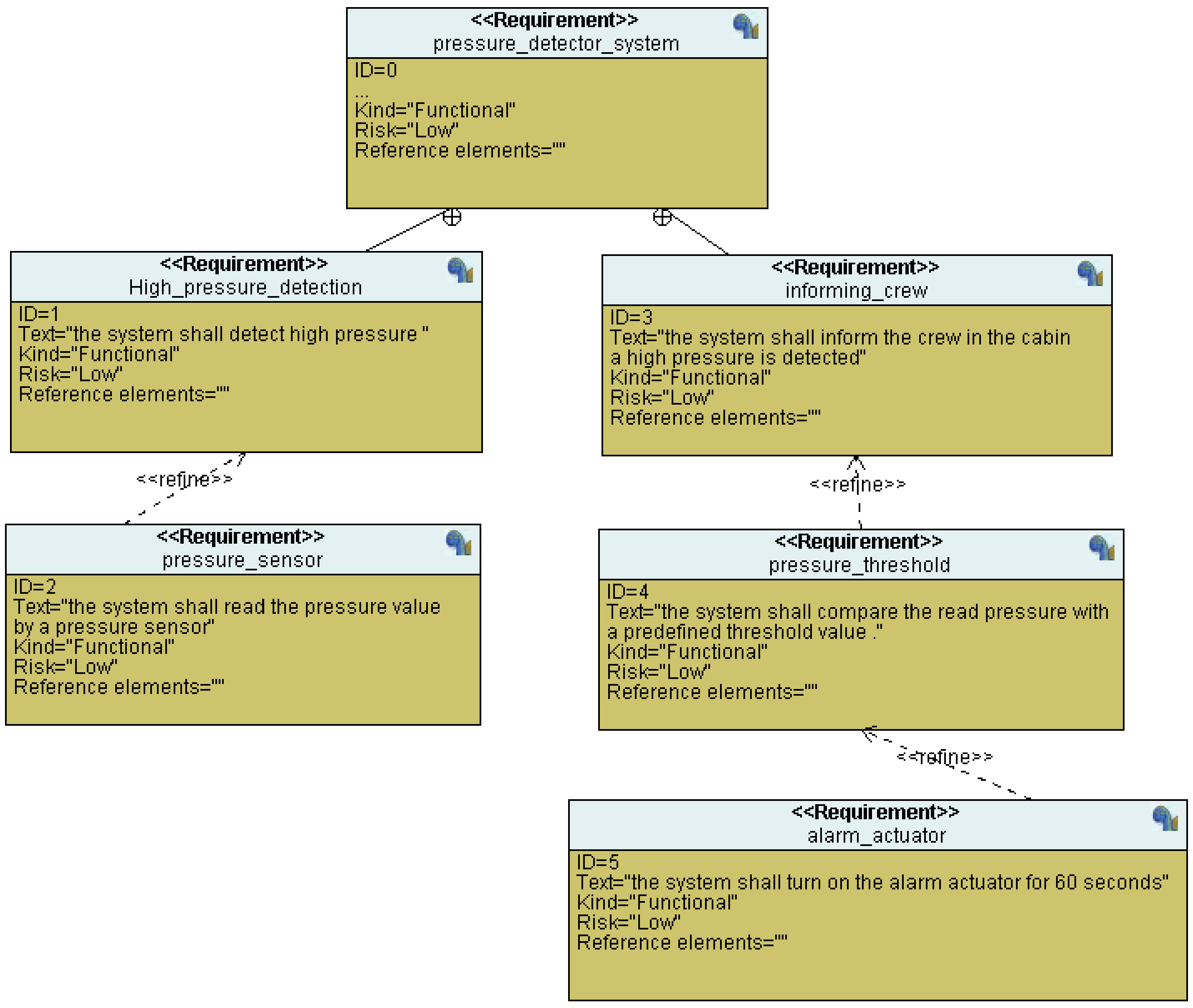
1- Use Case :

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

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.

2- Requirement Diagram :



<<Requirement>>
pressure_detector_system
ID=0
...
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
High_pressure_detection
ID=1
Text="the system shall detect high pressure "
Kind="Functional"
Risk="Low"
Reference elements=""

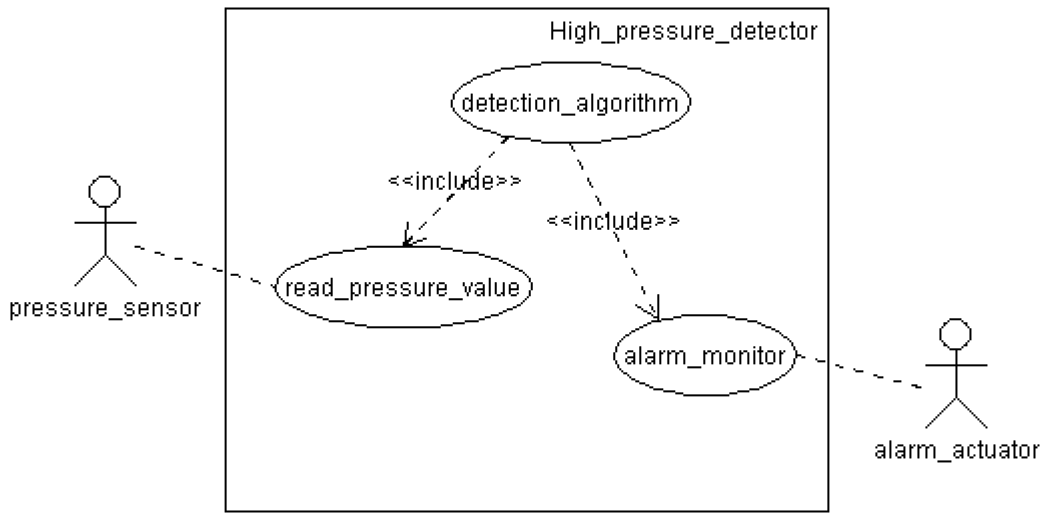
<<Requirement>>
informing_crew
ID=3
Text="the system shall inform the crew in the cabin a high pressure is detected"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
pressure_sensor
ID=2
Text="the system shall read the pressure value by a pressure sensor"
Kind="Functional"
Risk="Low"
Reference elements=""

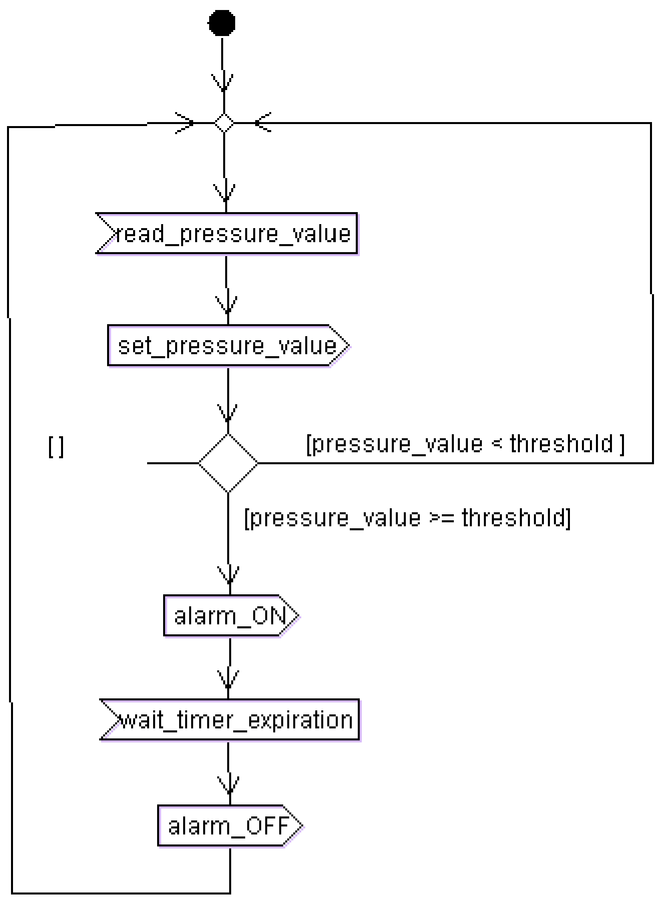
<<Requirement>>
pressure_threshold
ID=4
Text="the system shall compare the read pressure with a predefined threshold value ."
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
alarm_actuator
ID=5
Text="the system shall turn on the alarm actuator for 60 seconds"
Kind="Functional"
Risk="Low"
Reference elements=""

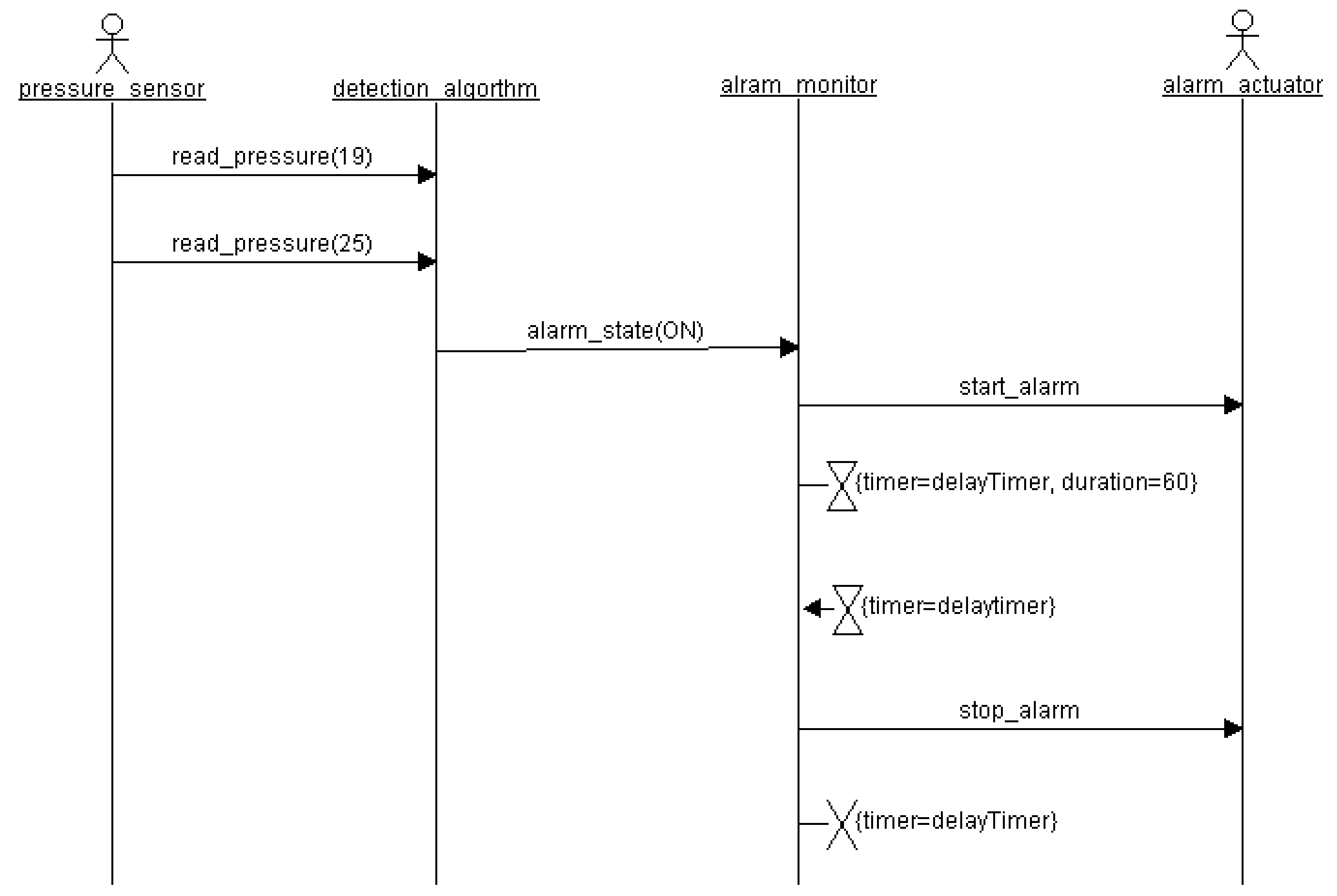
3- Use case Diagram:



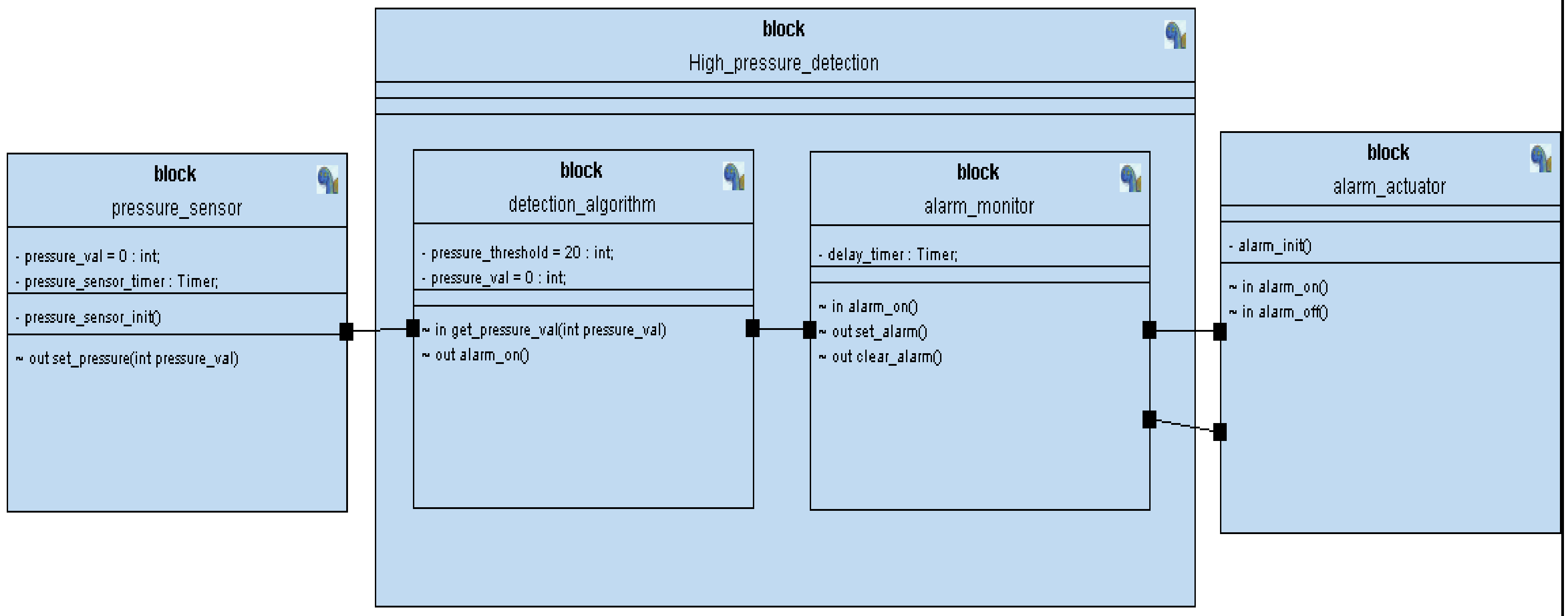
4- Activity Diagram:



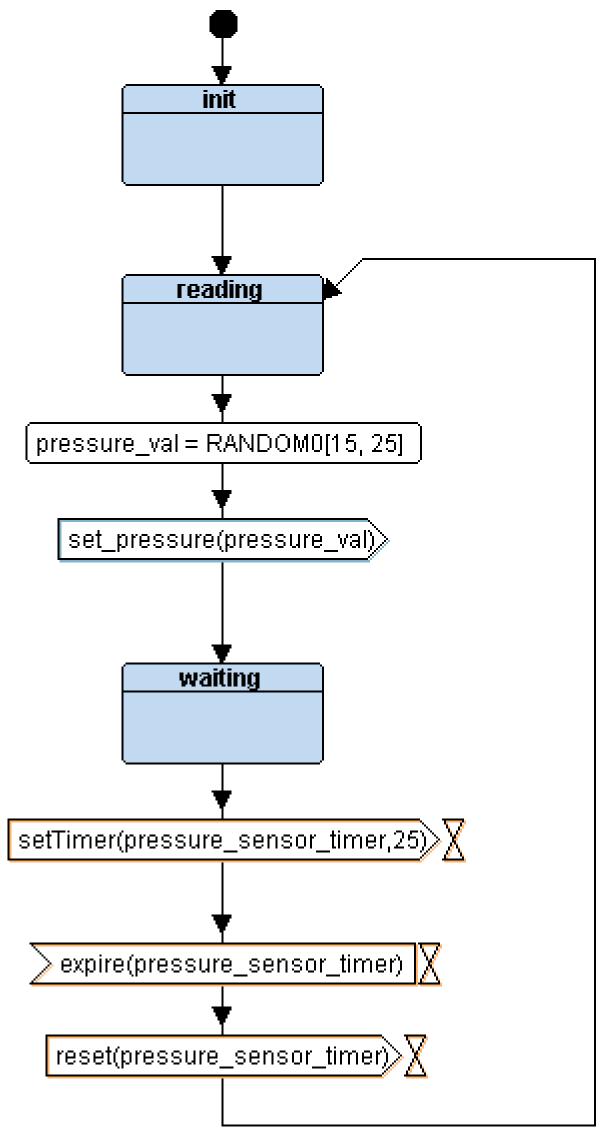
4- Sequence Diagram:



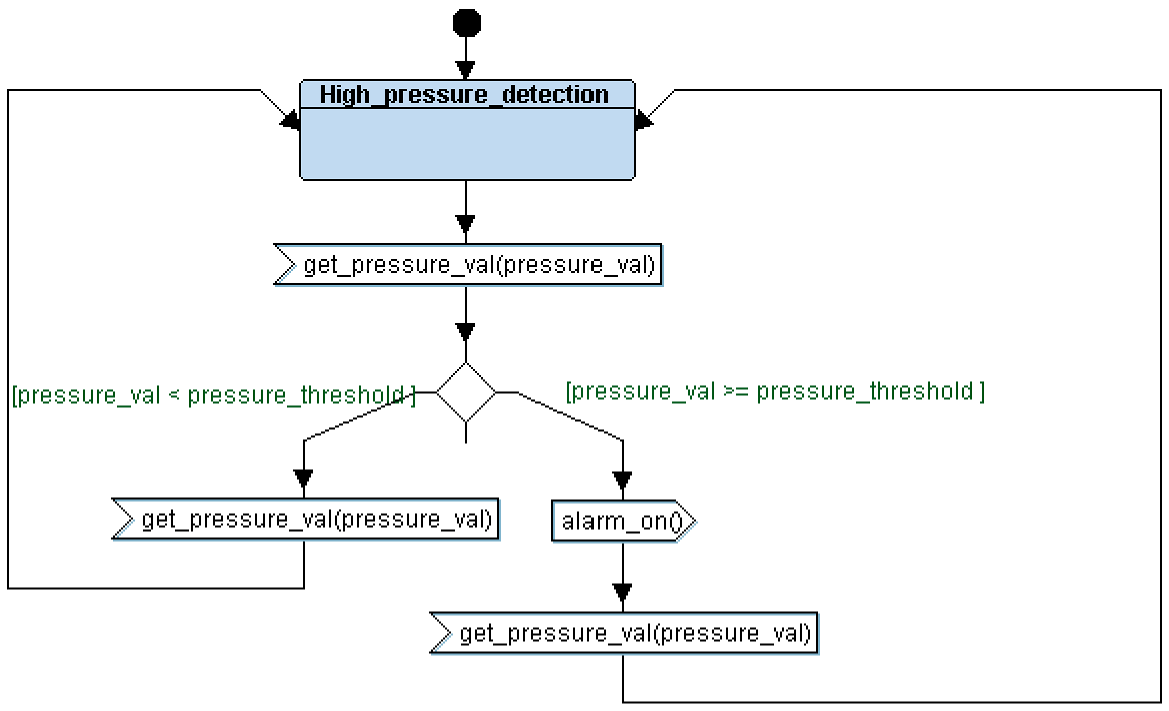
5- System Design:



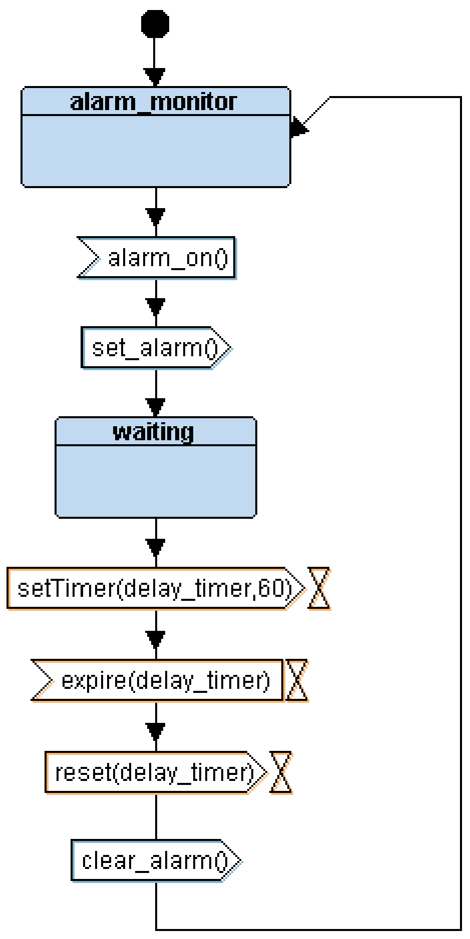
Pressure_sensor state diagram:


init
reading
pressure_val = RANDOM0[15, 25]
set_pressure(pressure_val)
waiting
setTimer(pressure_sensor_timer,25)
expire(pressure_sensor_timer)
reset(pressure_sensor_timer)

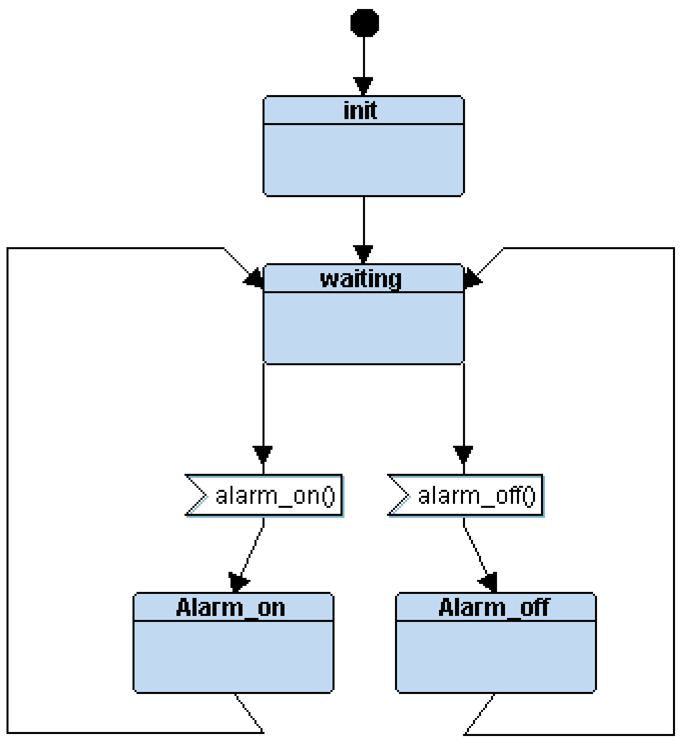
detection_algorithm state diagram:


High_pressure_detection
get_pressure_val(pressure_val)
[pressure_val < pressure_threshold]
[pressure_val >= pressure_threshold]
get_pressure_val(pressure_val)
alarm_on()
get_pressure_val(pressure_val)

alarm_monitor state diagram:



alarm_actuator state diagram:



6- Codes:

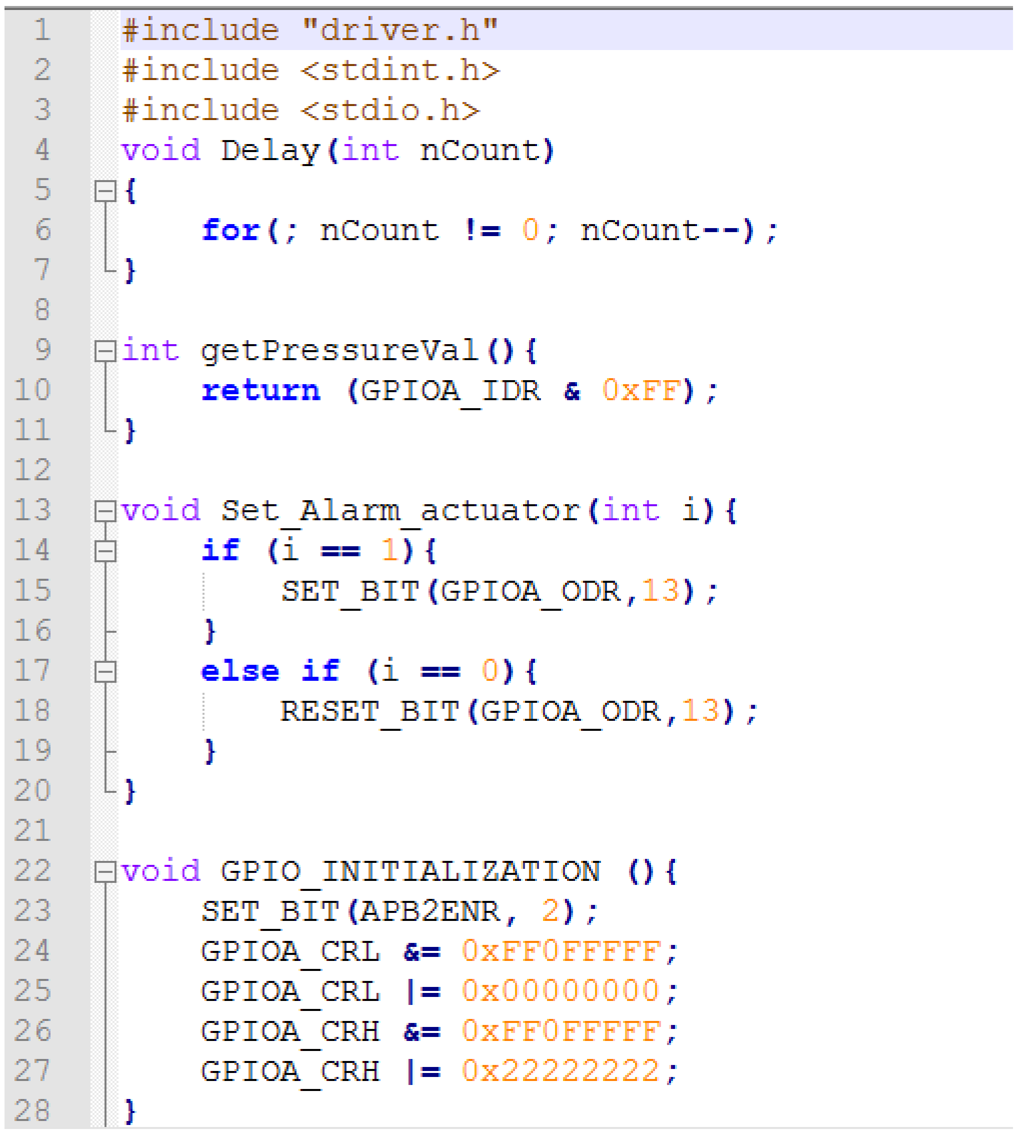
main.c:

```c
#include <stdint.h>
#include "driver.h"
#include "pressure_sensor.h"
#include "detection_algorithm.h"
#include "alarm_monitor.h"

int main(void )
{

    GPIO_INITIALIZATION ();

    while(1)
    {
        Pressure_state();
        Detect_state();
        Alarm_state();
    }


}
```

driver.h:

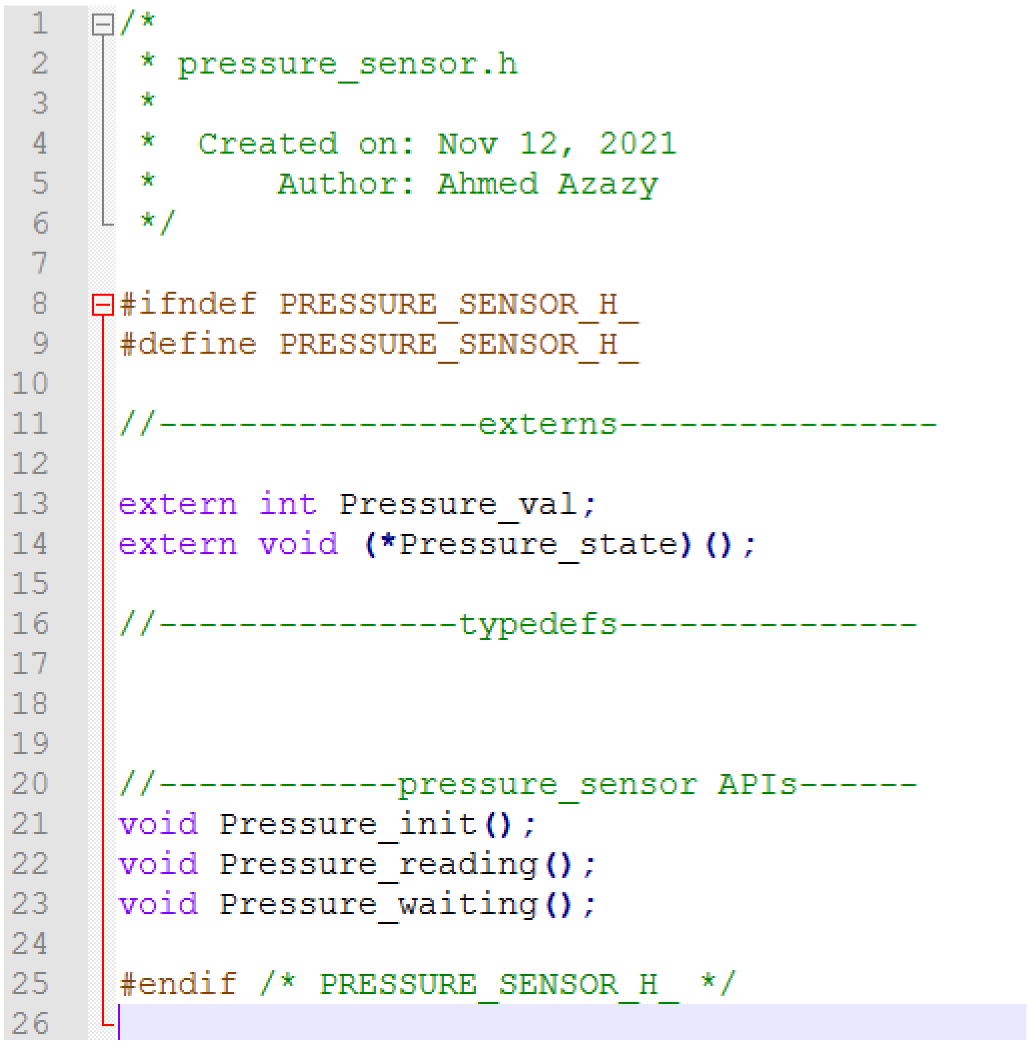
```c
#include <stdio.h>

#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 ();
```

driver.c:



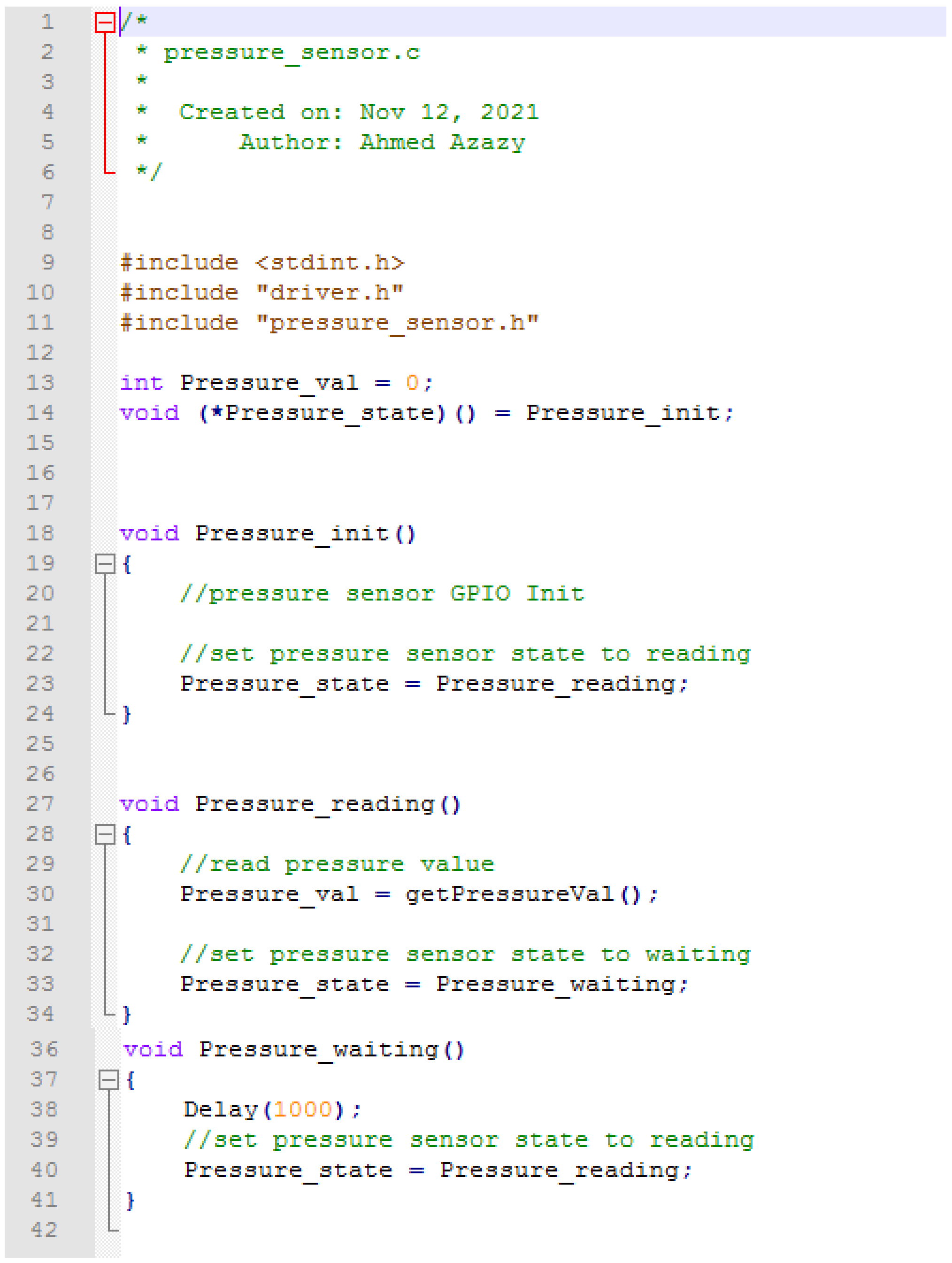
```c
#include "driver.h"
#include <stdint.h>
#include <stdio.h>
void Delay(int nCount)
{
	for(; nCount != 0; nCount--);
}

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;
}
```

pressure_sensor.h:

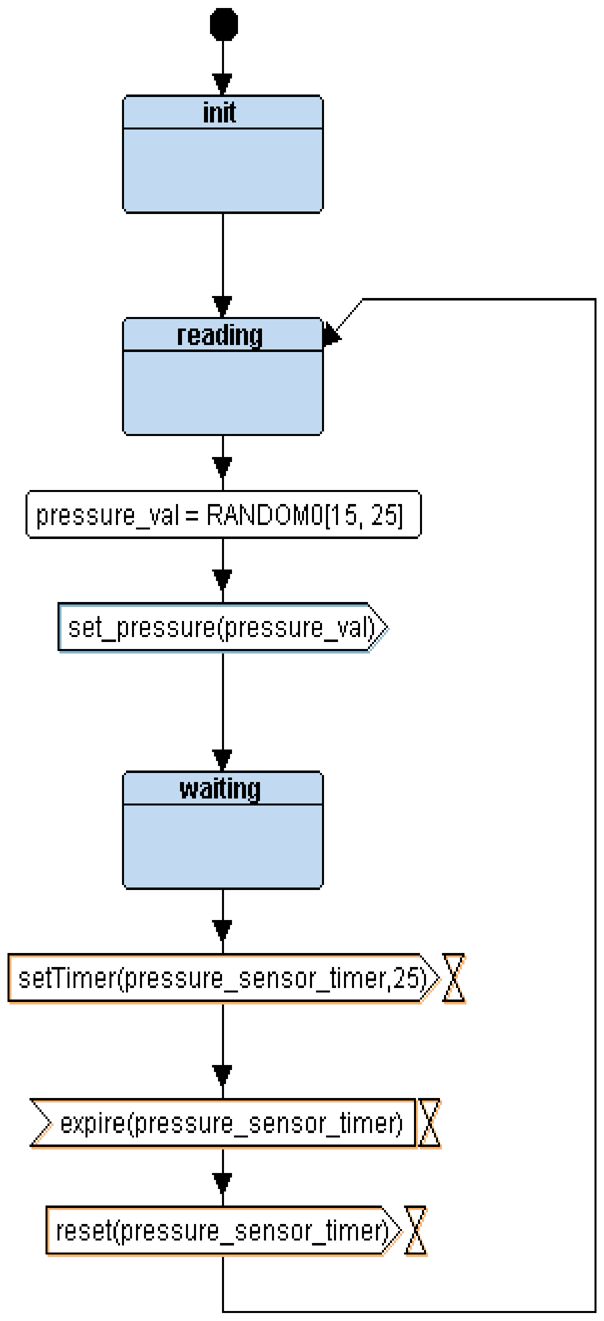


```c
/*
 * pressure_sensor.h
 *
 *  Created on: Nov 12, 2021
 *      Author: Ahmed Azazy
 */

#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_

//----------------externs----------------

extern int Pressure_val;
extern void (*Pressure_state)();

//---------------typedefs---------------



//------------pressure_sensor APIs------
void Pressure_init();
void Pressure_reading();
void Pressure_waiting();

#endif /* PRESSURE_SENSOR_H_ */

```

pressure_sensor.c:

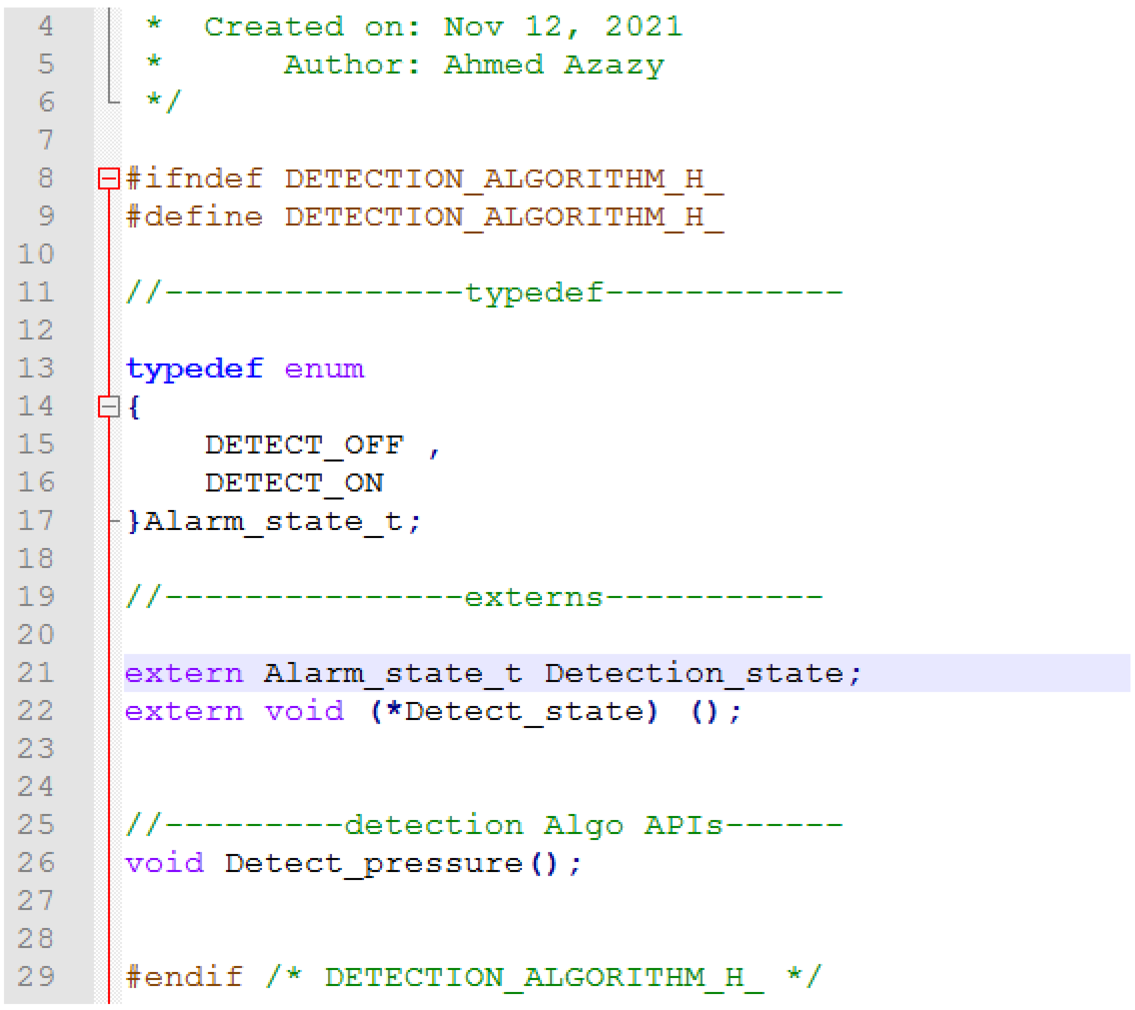


```c
/*
 * pressure_sensor.c
 *
 *  Created on: Nov 12, 2021
 *      Author: Ahmed Azazy
 */


#include <stdint.h>
#include "driver.h"
#include "pressure_sensor.h"

int Pressure_val = 0;
void (*Pressure_state)() = Pressure_init;



void Pressure_init()
{
    //pressure sensor GPIO Init

    //set pressure sensor state to reading
    Pressure_state = Pressure_reading;
}


void Pressure_reading()
{
    //read pressure value
    Pressure_val = getPressureVal();

    //set pressure sensor state to waiting
    Pressure_state = Pressure_waiting;
}
void Pressure_waiting()
{
    Delay(1000);
    //set pressure sensor state to reading
    Pressure_state = Pressure_reading;
}

```



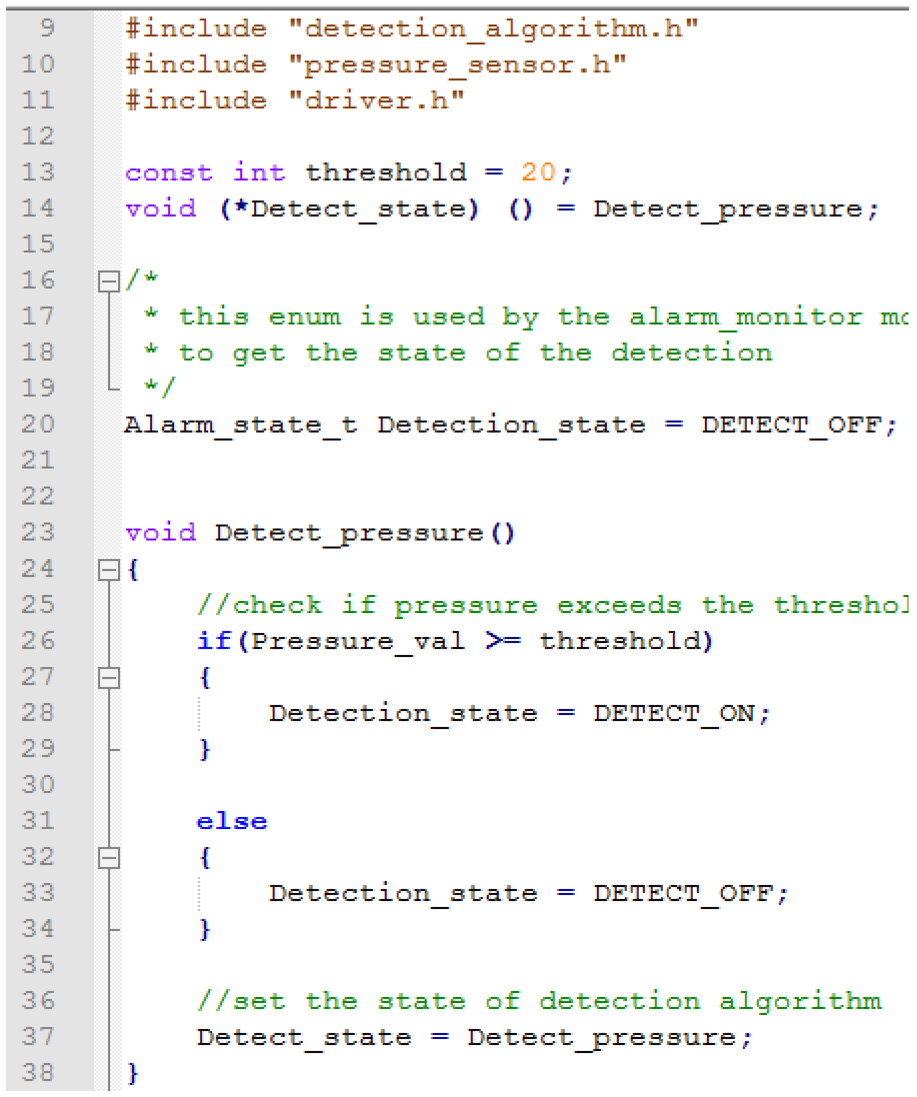
init
reading
pressure_val = RANDOM0[15, 25]
set_pressure(pressure_val)
waiting
setTimer(pressure_sensor_timer,25)
expire(pressure_sensor_timer)
reset(pressure_sensor_timer)

detection_algorithm.h:

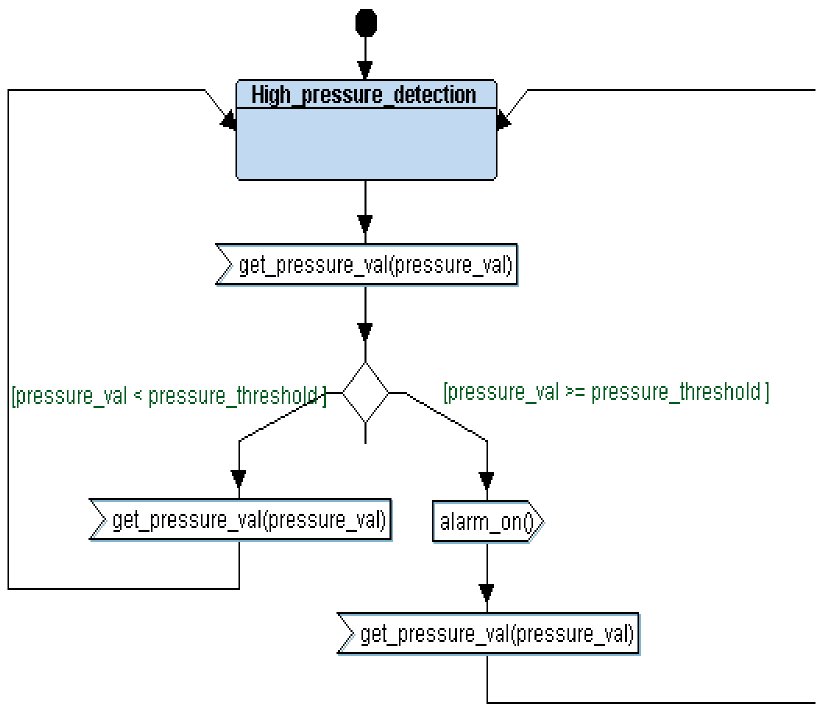


```c
 *  Created on: Nov 12, 2021
 *      Author: Ahmed Azazy
 */

#ifndef DETECTION_ALGORITHM_H_
#define DETECTION_ALGORITHM_H_

//---------------typedef------------

typedef enum
{
    DETECT_OFF ,
    DETECT_ON
}Alarm_state_t;

//---------------externs-----------

extern Alarm_state_t Detection_state;
extern void (*Detect_state) ();


//---------detection Algo APIs------
void Detect_pressure();


#endif /* DETECTION_ALGORITHM_H_ */
```

detection_algorithm.c:

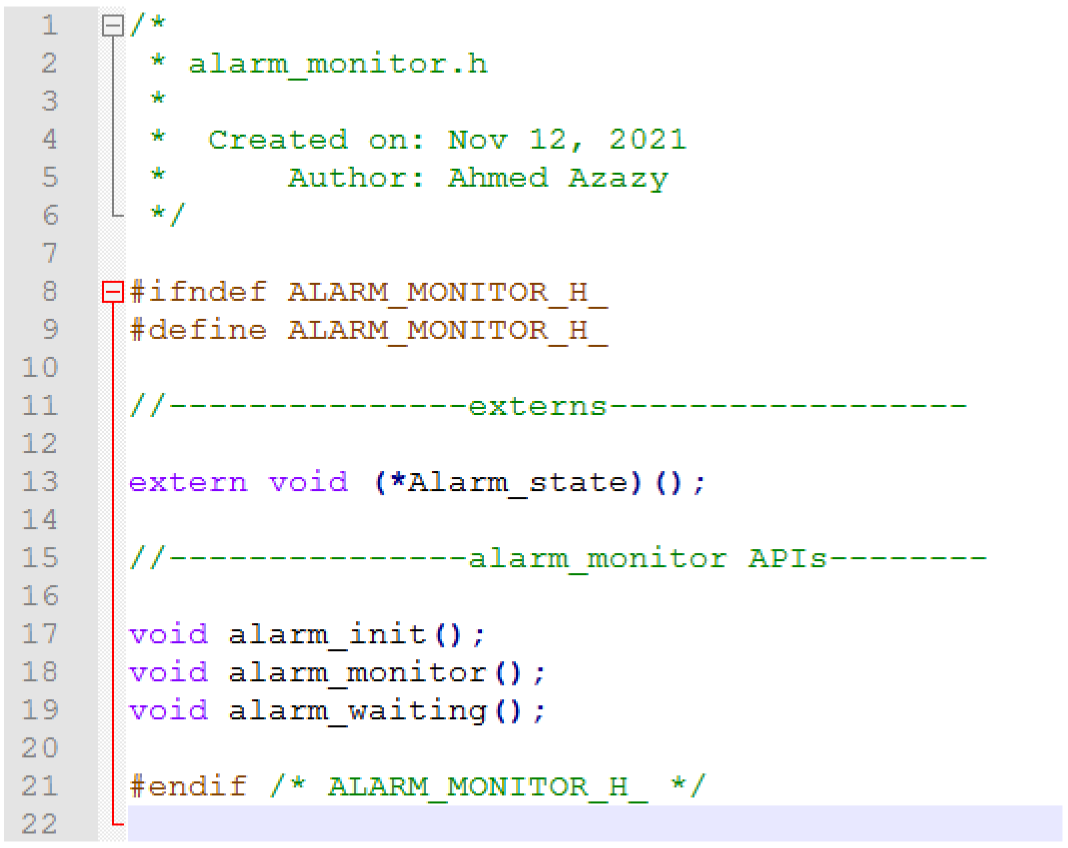


```c
#include "detection_algorithm.h"
#include "pressure_sensor.h"
#include "driver.h"

const int threshold = 20;
void (*Detect_state) () = Detect_pressure;

/*
 * this enum is used by the alarm_monitor mo
 * to get the state of the detection
 */
Alarm_state_t Detection_state = DETECT_OFF;


void Detect_pressure()
{
    //check if pressure exceeds the threshol
    if(Pressure_val >= threshold)
    {
        Detection_state = DETECT_ON;
    }

    else
    {
        Detection_state = DETECT_OFF;
    }

    //set the state of detection algorithm
    Detect_state = Detect_pressure;
}
```



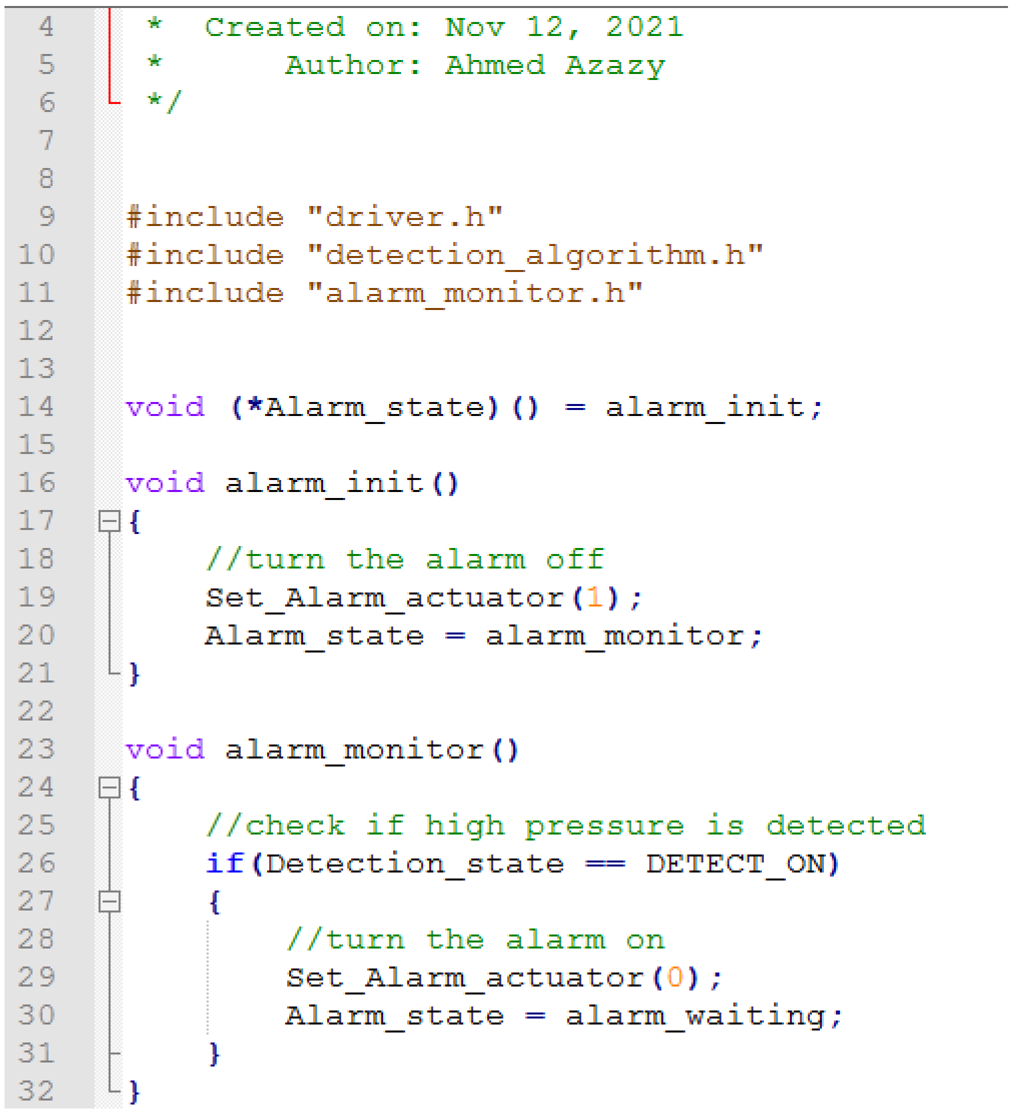
High_pressure_detection
get_pressure_val(pressure_val)
[pressure_val < pressure_threshold]
[pressure_val >= pressure_threshold]
get_pressure_val(pressure_val)
alarm_on()
get_pressure_val(pressure_val)

alarm_monitor.h:

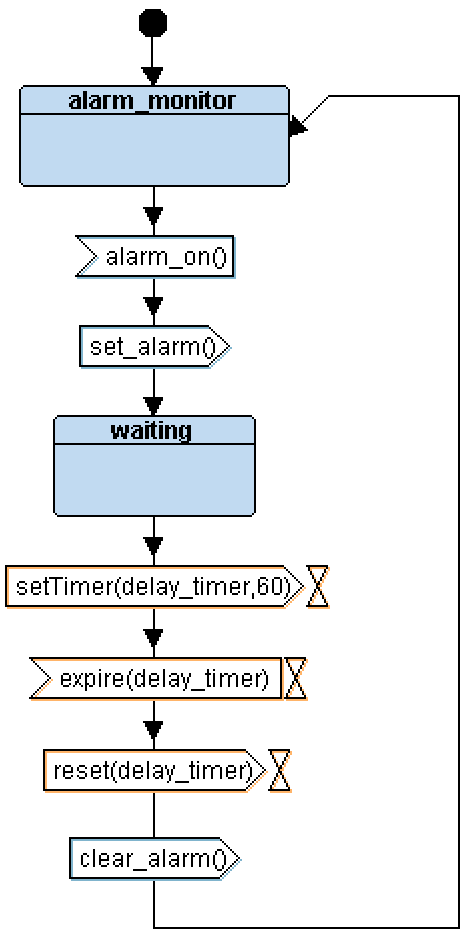


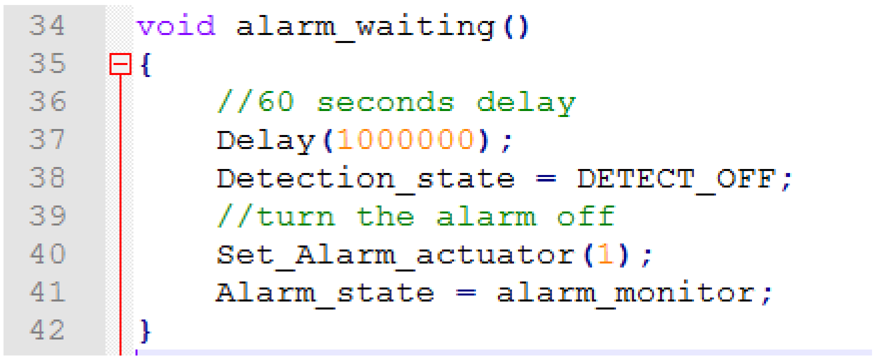
```c
/*
 * alarm_monitor.h
 *
 *  Created on: Nov 12, 2021
 *      Author: Ahmed Azazy
 */

#ifndef ALARM_MONITOR_H_
#define ALARM_MONITOR_H_

//---------------externs-----------------

extern void (*Alarm_state)();

//---------------alarm_monitor APIs--------

void alarm_init();
void alarm_monitor();
void alarm_waiting();

#endif /* ALARM_MONITOR_H_ */

```

alarm_monitor.c:



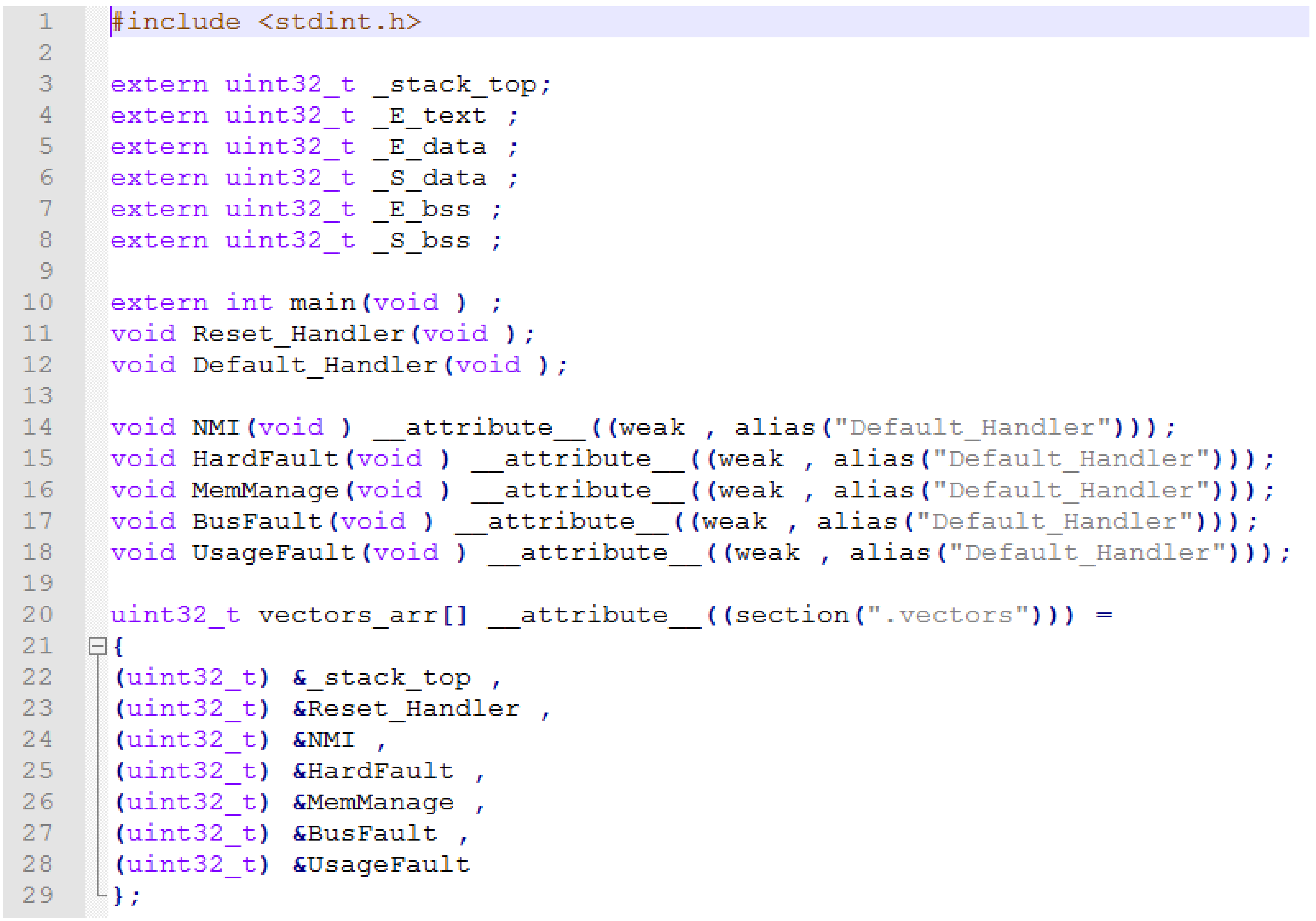
```c
 *  Created on: Nov 12, 2021
 *      Author: Ahmed Azazy
 */


#include "driver.h"
#include "detection_algorithm.h"
#include "alarm_monitor.h"


void (*Alarm_state)() = alarm_init;

void alarm_init()
{
    //turn the alarm off
    Set_Alarm_actuator(1);
    Alarm_state = alarm_monitor;
}

void alarm_monitor()
{
    //check if high pressure is detected
    if(Detection_state == DETECT_ON)
    {
        //turn the alarm on
        Set_Alarm_actuator(0);
        Alarm_state = alarm_waiting;
    }
}
```



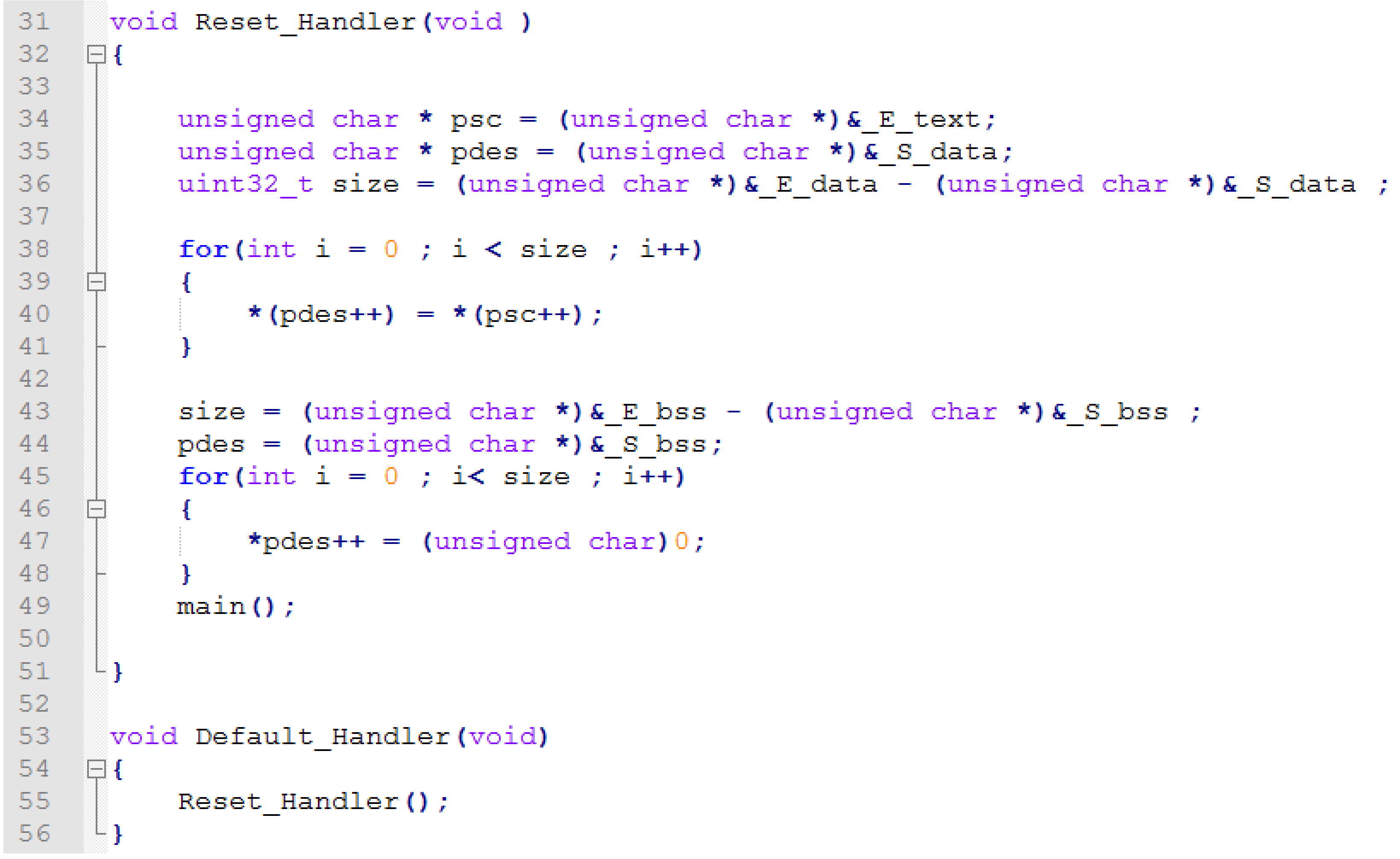


```c
void alarm_waiting()
{
    //60 seconds delay
    Delay(1000000);
    Detection_state = DETECT_OFF;
    //turn the alarm off
    Set_Alarm_actuator(1);
    Alarm_state = alarm_monitor;
}
```

Startup.c:



```c
#include <stdint.h>

extern uint32_t _stack_top;
extern uint32_t _E_text ;
extern uint32_t _E_data ;
extern uint32_t _S_data ;
extern uint32_t _E_bss ;
extern uint32_t _S_bss ;

extern int main(void ) ;
void Reset_Handler(void );
void Default_Handler(void );

void NMI(void ) __attribute__((weak , alias("Default_Handler")));
void HardFault(void ) __attribute__((weak , alias("Default_Handler")));
void MemManage(void ) __attribute__((weak , alias("Default_Handler")));
void BusFault(void ) __attribute__((weak , alias("Default_Handler")));
void UsageFault(void ) __attribute__((weak , alias("Default_Handler")));

uint32_t vectors_arr[] __attribute__((section(".vectors"))) =
{
(uint32_t) &_stack_top ,
(uint32_t) &Reset_Handler ,
(uint32_t) &NMI ,
(uint32_t) &HardFault ,
(uint32_t) &MemManage ,
(uint32_t) &BusFault ,
(uint32_t) &UsageFault
};
```

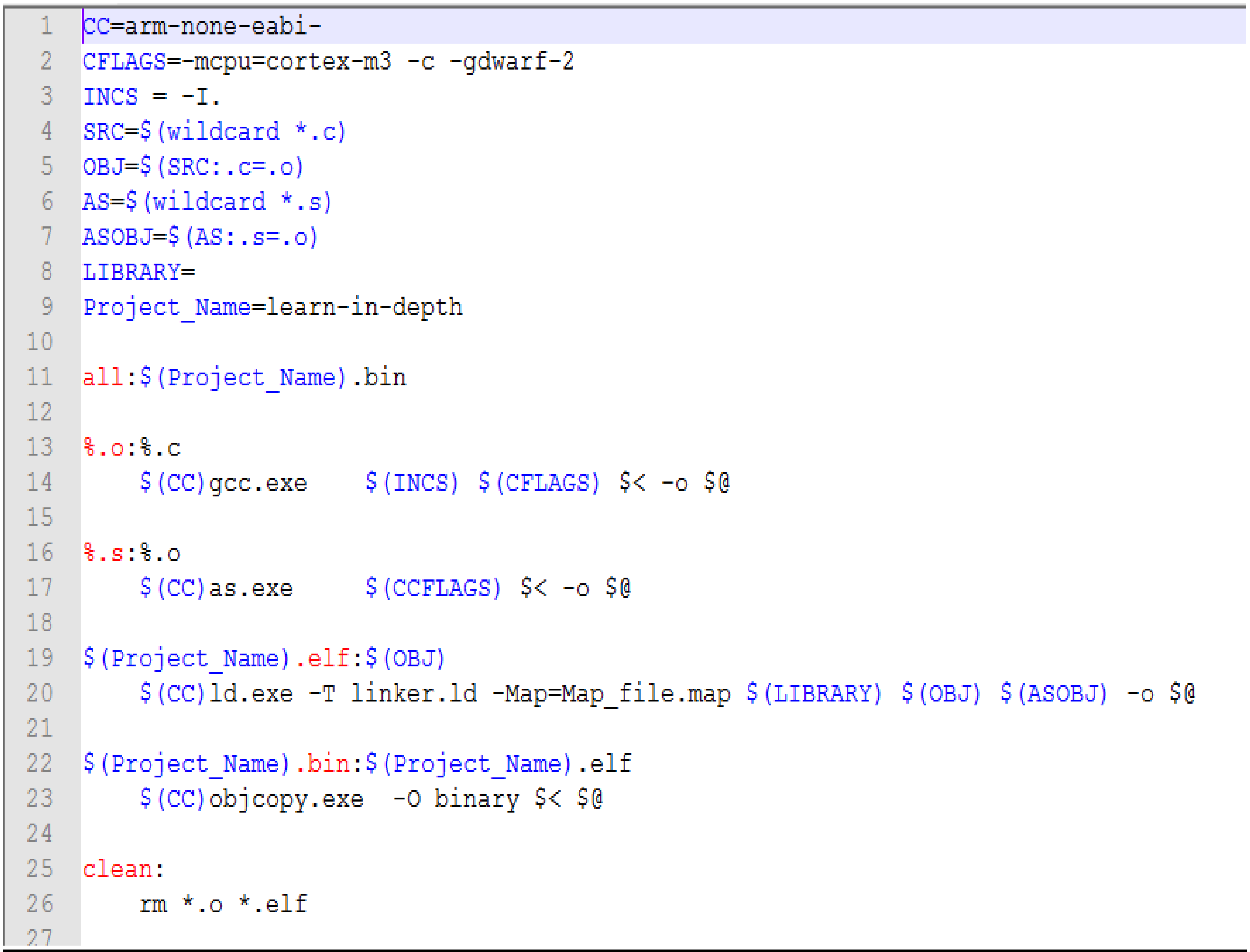


```c
void Reset_Handler(void )
{

    unsigned char * psc = (unsigned char *)&_E_text;
    unsigned char * pdes = (unsigned char *)&_S_data;
    uint32_t size = (unsigned char *)&_E_data - (unsigned char *)&_S_data ;

    for(int i = 0 ; i < size ; i++)
    {
        *(pdes++) = *(psc++);
    }

    size = (unsigned char *)&_E_bss - (unsigned char *)&_S_bss ;
    pdes = (unsigned char *)&_S_bss;
    for(int i = 0 ; i< size ; i++)
    {
        *pdes++ = (unsigned char)0;
    }
    main();

}

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

Linker_script.ld:

```
MEMORY
{
    FLASH (rx) : ORIGIN = 0x08000000 , LENGTH = 128K
    SRAM (rwx) : ORIGIN = 0x20000000 , LENGTH = 20K
}

SECTIONS
{
    .text :
    {
        startup.o(.vextors)
        *(.vectors*)
        *(.text)
        *(.text*)
        *(.rodata)
        . = ALIGN(4) ;
        _E_text = . ;
    } > FLASH

    .data :
    {
        _S_data = .;
        *(.data)
        *(.data*)
        . = ALIGN(4) ;
        _E_data = . ;
    } > SRAM AT> FLASH
    .bss :
    {
        _S_bss = . ;
        *(.bss)
        *(.bss*)
        . = ALIGN(4) ;
        _E_bss = . ;

        . = . + 0x1000 ;
        _stack_top = . ;
    } > SRAM
}
```

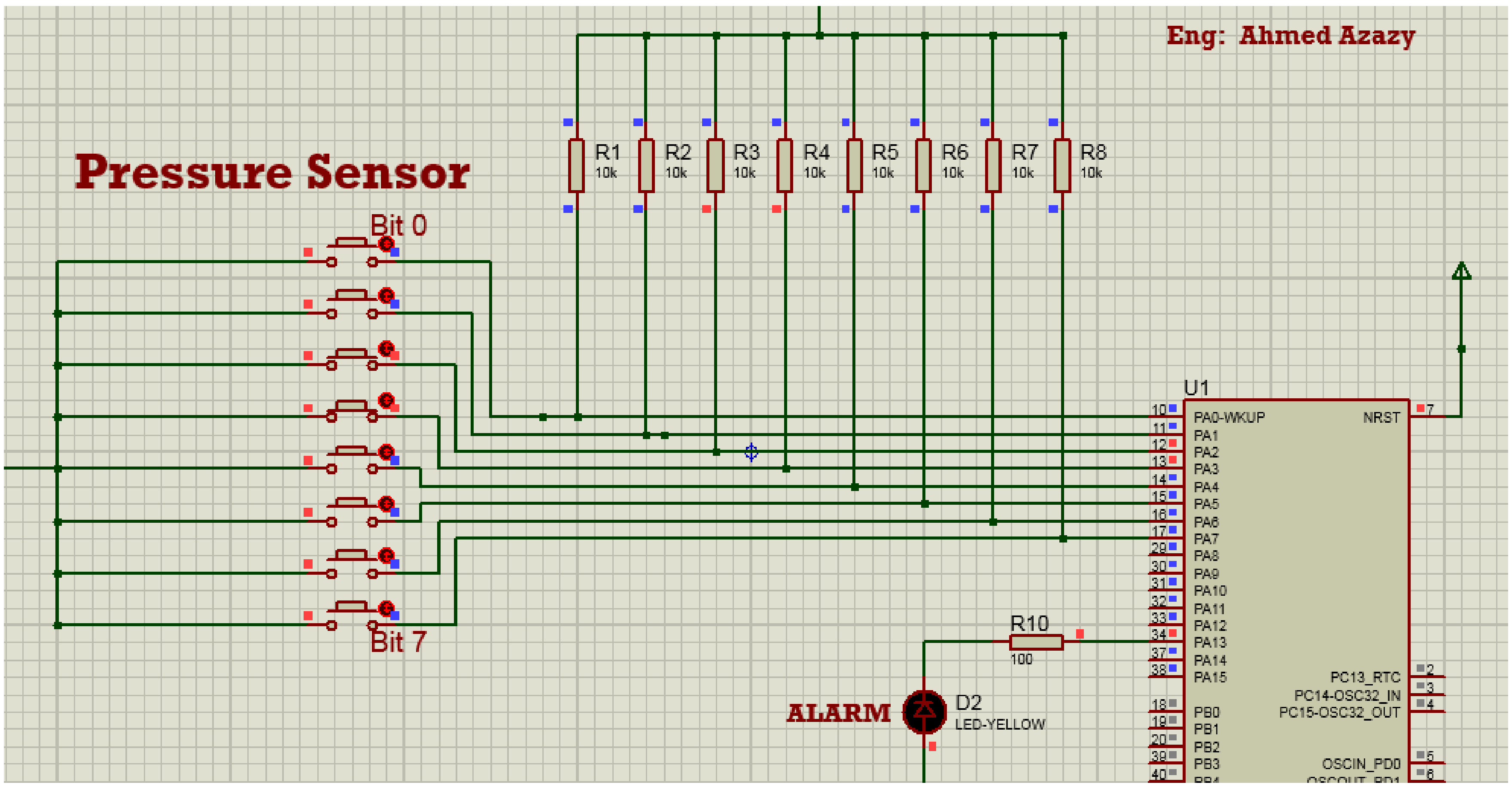
Makefile:



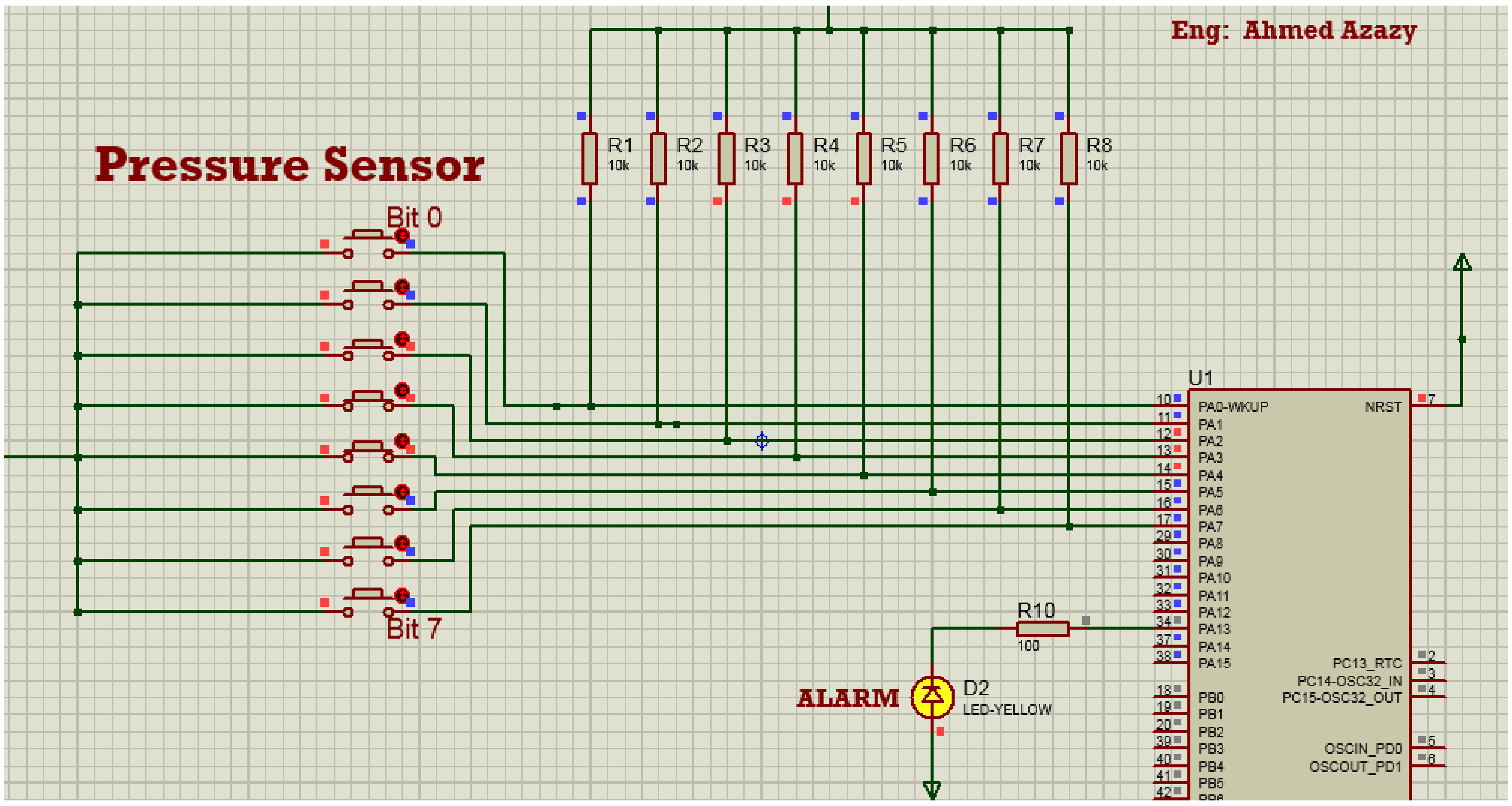
```makefile
CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -c -gdwarf-2
INCS = -I.
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
AS=$(wildcard *.s)
ASOBJ=$(AS:.s=.o)
LIBRARY=
Project_Name=learn-in-depth

all:$(Project_Name).bin

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

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

$(Project_Name).elf:$(OBJ)
	$(CC)ld.exe -T linker.ld -Map=Map_file.map $(LIBRARY) $(OBJ) $(ASOBJ) -o $@

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

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

7- Simulation Results:

Pressure = 12 , then pressure is below the threshold(20 bar).
So , **the Alarm is OFF**.

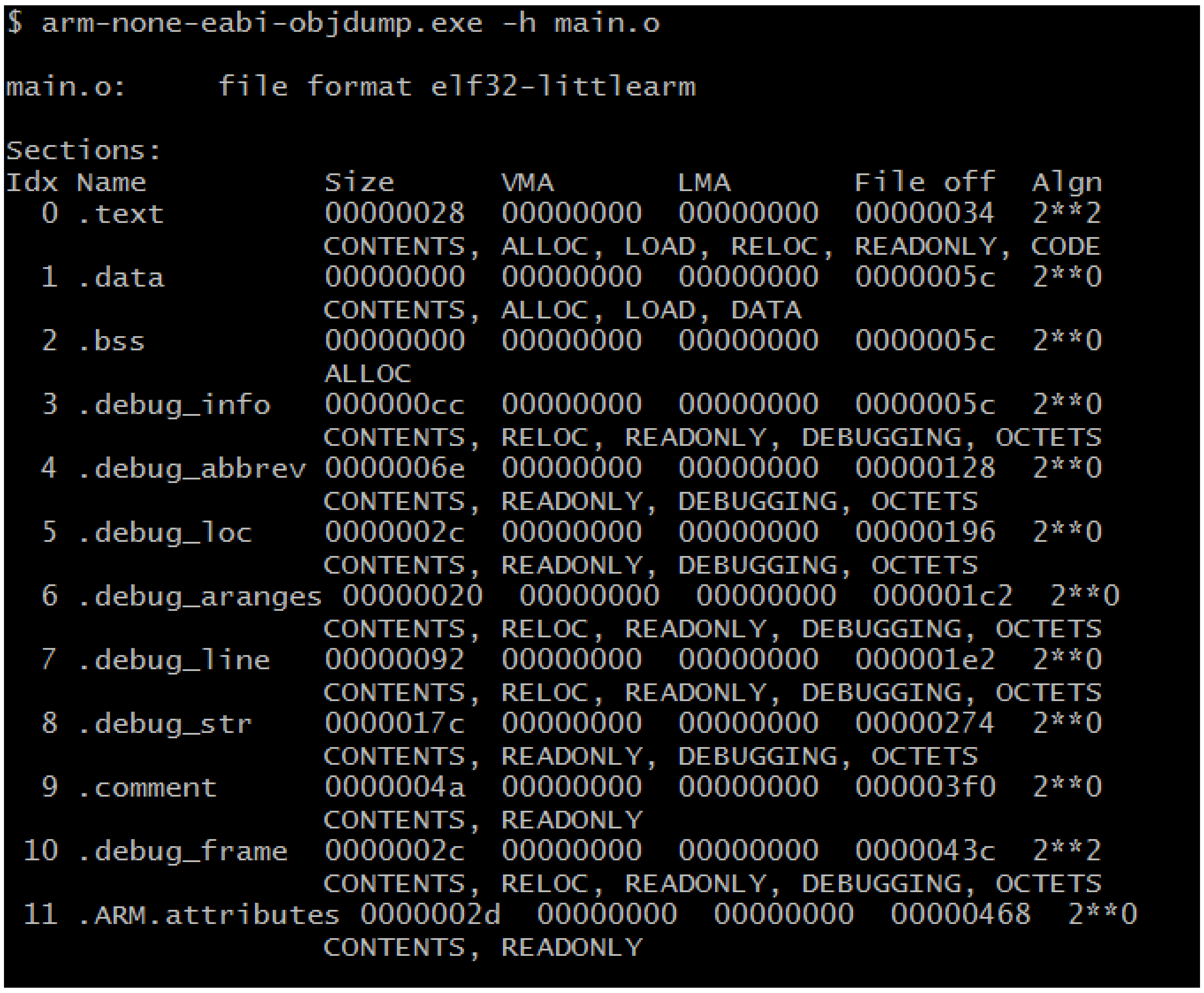


When **Pressure = 28** , then pressure is above the threshold(20 bar).
So , **the Alarm is ON**.



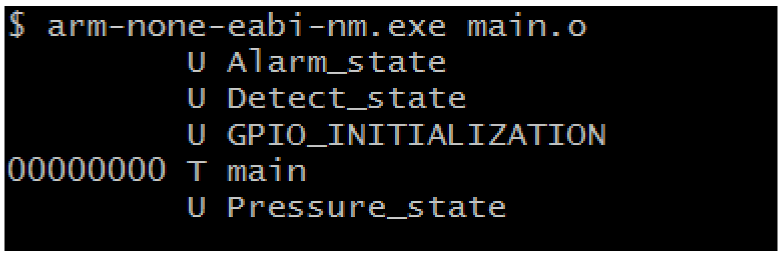
8- Software Analysis:

main.o sections:



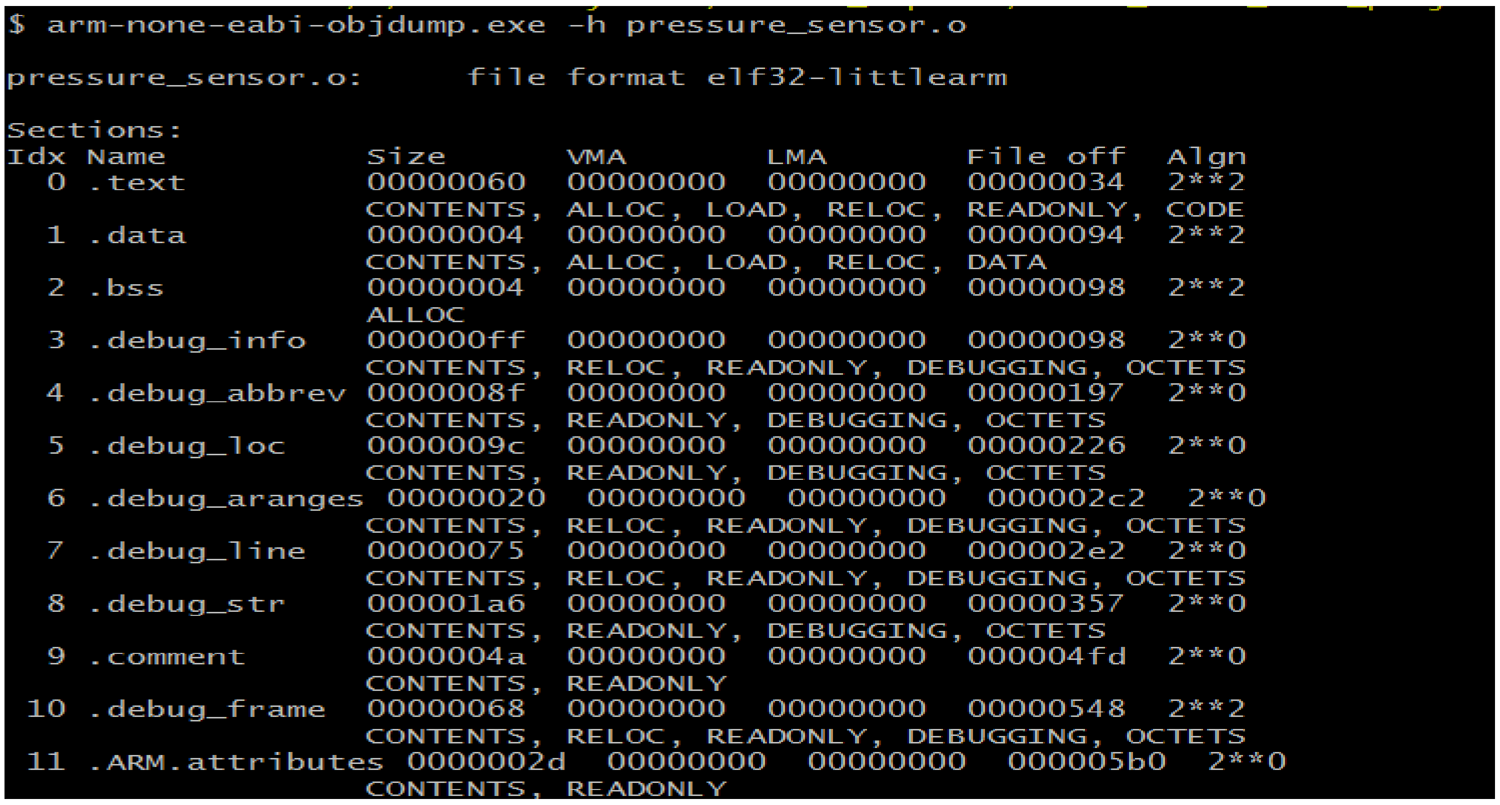
```
$ arm-none-eabi-objdump.exe -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000028  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000005c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000005c  2**0
                  ALLOC
  3 .debug_info   000000cc  00000000  00000000  0000005c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  4 .debug_abbrev 0000006e  00000000  00000000  00000128  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  5 .debug_loc    0000002c  00000000  00000000  00000196  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  6 .debug_aranges 00000020  00000000  00000000  000001c2  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  7 .debug_line   00000092  00000000  00000000  000001e2  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  8 .debug_str    0000017c  00000000  00000000  00000274  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  9 .comment      0000004a  00000000  00000000  000003f0  2**0
                  CONTENTS, READONLY
 10 .debug_frame  0000002c  00000000  00000000  0000043c  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 11 .ARM.attributes 0000002d  00000000  00000000  00000468  2**0
                  CONTENTS, READONLY
```

main.o symbols:



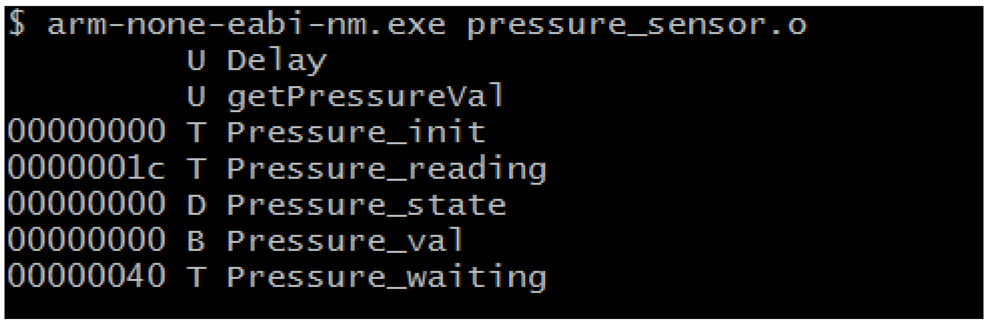
```
$ arm-none-eabi-nm.exe main.o
         U Alarm_state
         U Detect_state
         U GPIO_INITIALIZATION
00000000 T main
         U Pressure_state
```

pressure_sensor.o sections:



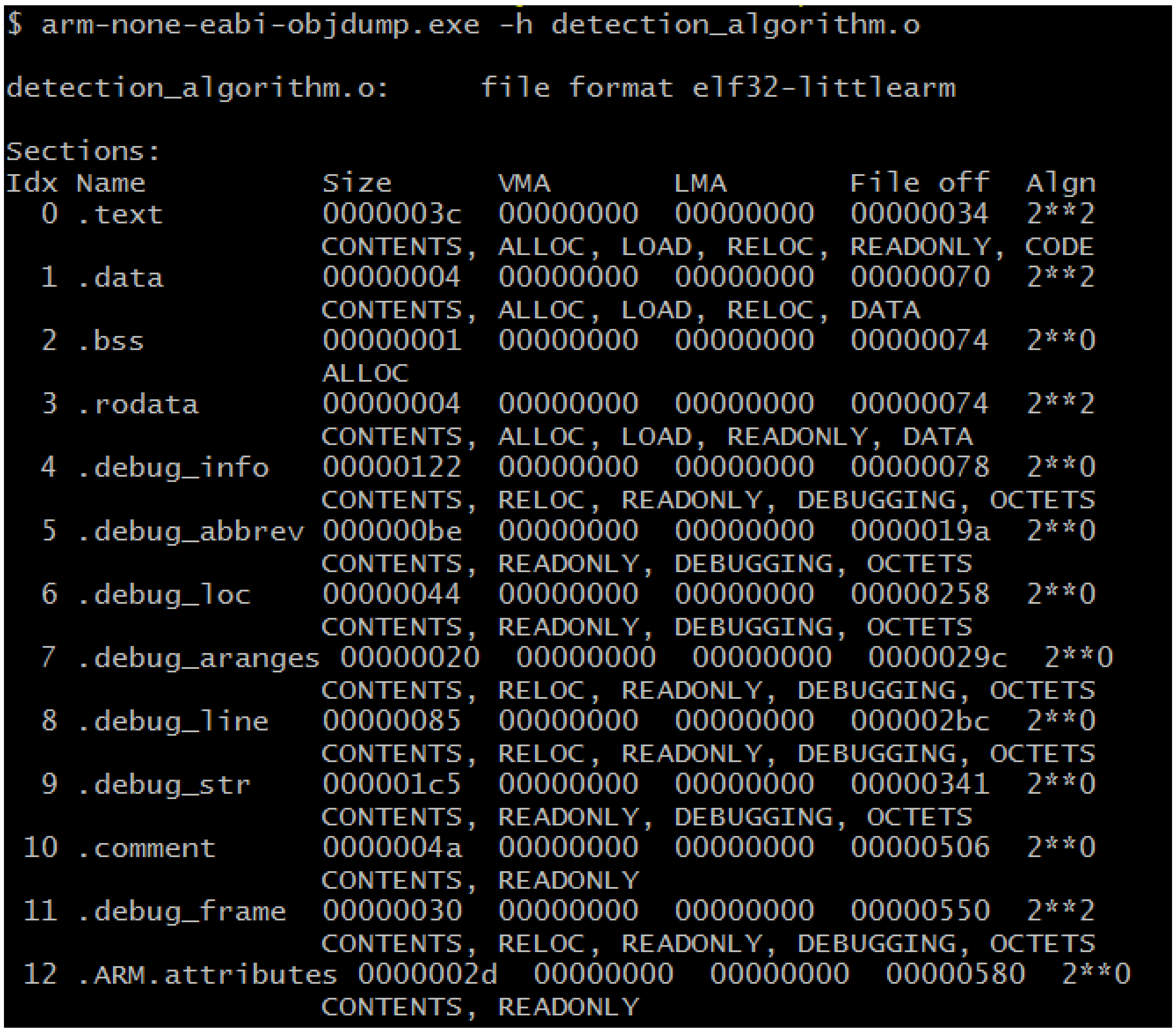
```
$ arm-none-eabi-objdump.exe -h pressure_sensor.o

pressure_sensor.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000060  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000004  00000000  00000000  00000094  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  2 .bss          00000004  00000000  00000000  00000098  2**2
                  ALLOC
  3 .debug_info   000000ff  00000000  00000000  00000098  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  4 .debug_abbrev 0000008f  00000000  00000000  00000197  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  5 .debug_loc    0000009c  00000000  00000000  00000226  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  6 .debug_aranges 00000020  00000000  00000000  000002c2  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  7 .debug_line   00000075  00000000  00000000  000002e2  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  8 .debug_str    000001a6  00000000  00000000  00000357  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  9 .comment      0000004a  00000000  00000000  000004fd  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000068  00000000  00000000  00000548  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 11 .ARM.attributes 0000002d  00000000  00000000  000005b0  2**0
                  CONTENTS, READONLY
```

pressure_sensor.o symbols:



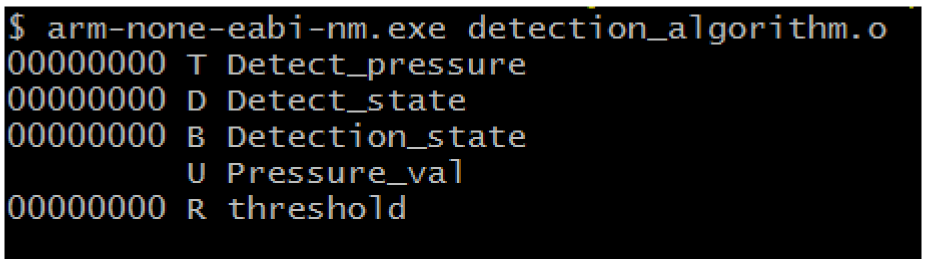
```
$ arm-none-eabi-nm.exe pressure_sensor.o
         U Delay
         U getPressureVal
00000000 T Pressure_init
0000001c T Pressure_reading
00000000 D Pressure_state
00000000 B Pressure_val
00000040 T Pressure_waiting
```

detection_algorithm.o sections:



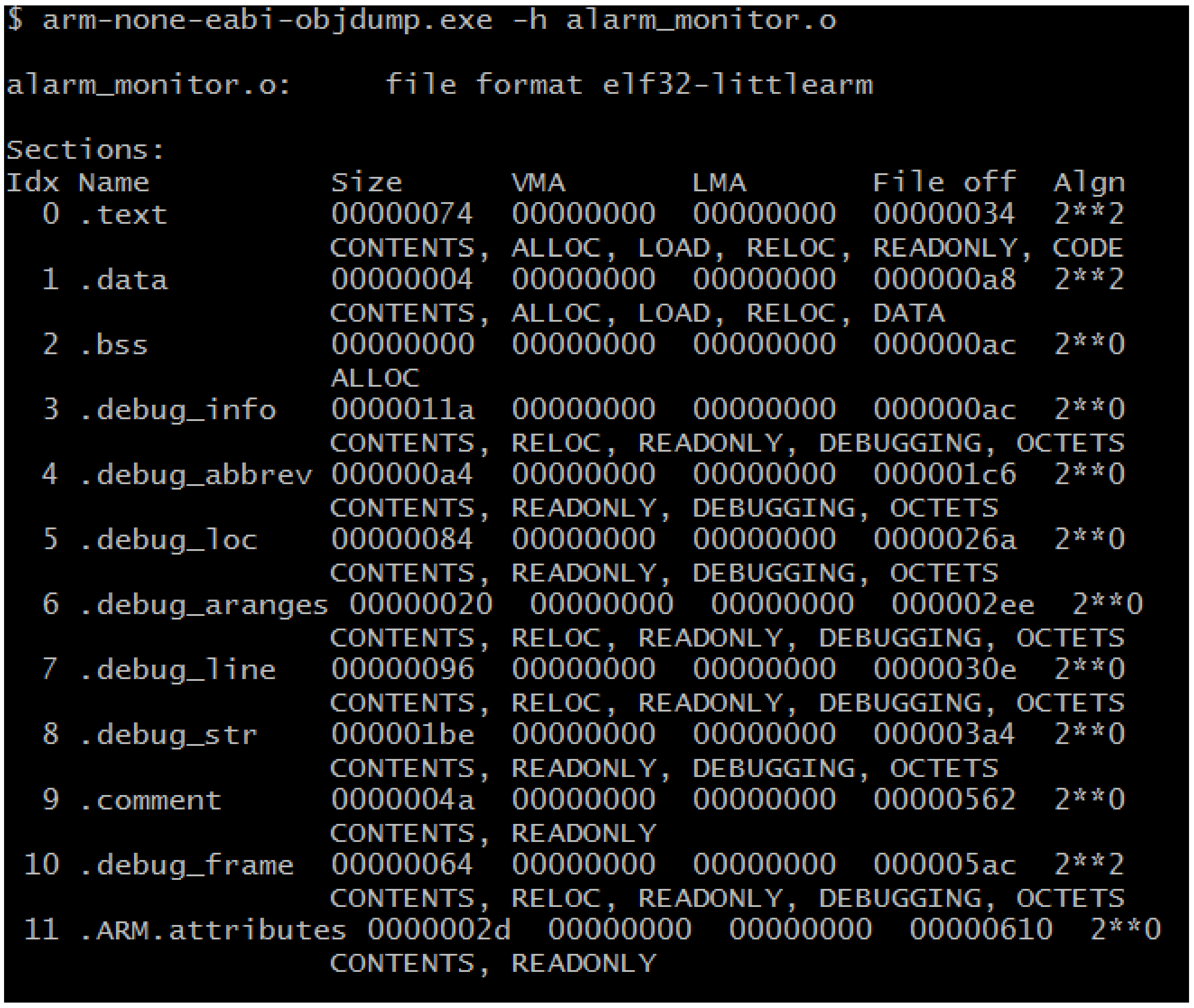
```
$ arm-none-eabi-objdump.exe -h detection_algorithm.o

detection_algorithm.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, RELOC, DATA
  2 .bss          00000001  00000000  00000000  00000074  2**0
                  ALLOC
  3 .rodata       00000004  00000000  00000000  00000074  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .debug_info   00000122  00000000  00000000  00000078  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  5 .debug_abbrev 000000be  00000000  00000000  0000019a  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  6 .debug_loc    00000044  00000000  00000000  00000258  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  7 .debug_aranges 00000020  00000000  00000000  0000029c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  8 .debug_line   00000085  00000000  00000000  000002bc  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  9 .debug_str    000001c5  00000000  00000000  00000341  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 10 .comment      0000004a  00000000  00000000  00000506  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000030  00000000  00000000  00000550  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 12 .ARM.attributes 0000002d  00000000  00000000  00000580  2**0
                  CONTENTS, READONLY
```

detection_algorithm.o symbols:



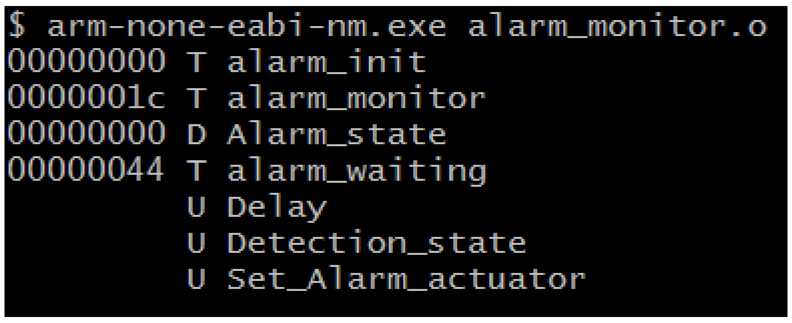
```
$ arm-none-eabi-nm.exe detection_algorithm.o
00000000 T Detect_pressure
00000000 D Detect_state
00000000 B Detection_state
         U Pressure_val
00000000 R threshold
```

alarm_monitor.o sections:



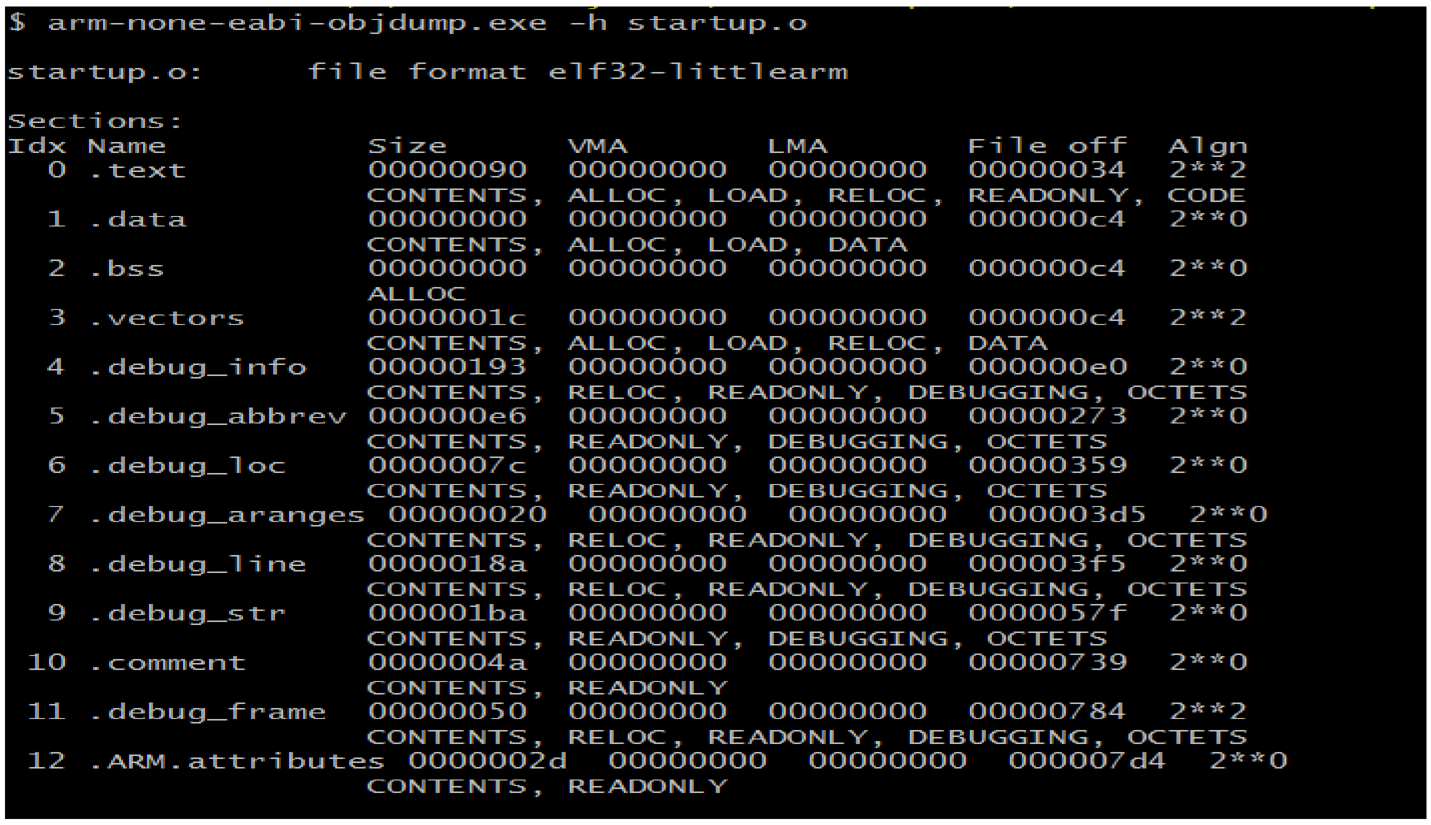
```
$ arm-none-eabi-objdump.exe -h alarm_monitor.o

alarm_monitor.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000074  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000004  00000000  00000000  000000a8  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  2 .bss          00000000  00000000  00000000  000000ac  2**0
                  ALLOC
  3 .debug_info   0000011a  00000000  00000000  000000ac  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  4 .debug_abbrev 000000a4  00000000  00000000  000001c6  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  5 .debug_loc    00000084  00000000  00000000  0000026a  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  6 .debug_aranges 00000020  00000000  00000000  000002ee  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  7 .debug_line   00000096  00000000  00000000  0000030e  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  8 .debug_str    000001be  00000000  00000000  000003a4  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  9 .comment      0000004a  00000000  00000000  00000562  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000064  00000000  00000000  000005ac  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 11 .ARM.attributes 0000002d  00000000  00000000  00000610  2**0
                  CONTENTS, READONLY
```

alarm_monitor.o symbols:



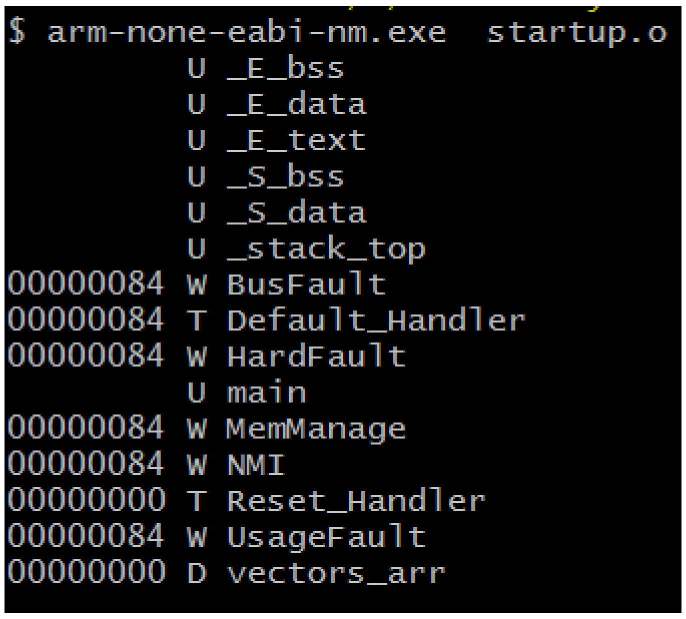
```
$ arm-none-eabi-nm.exe alarm_monitor.o
00000000 T alarm_init
0000001c T alarm_monitor
00000000 D Alarm_state
00000044 T alarm_waiting
         U Delay
         U Detection_state
         U Set_Alarm_actuator
```

startup.o sections:



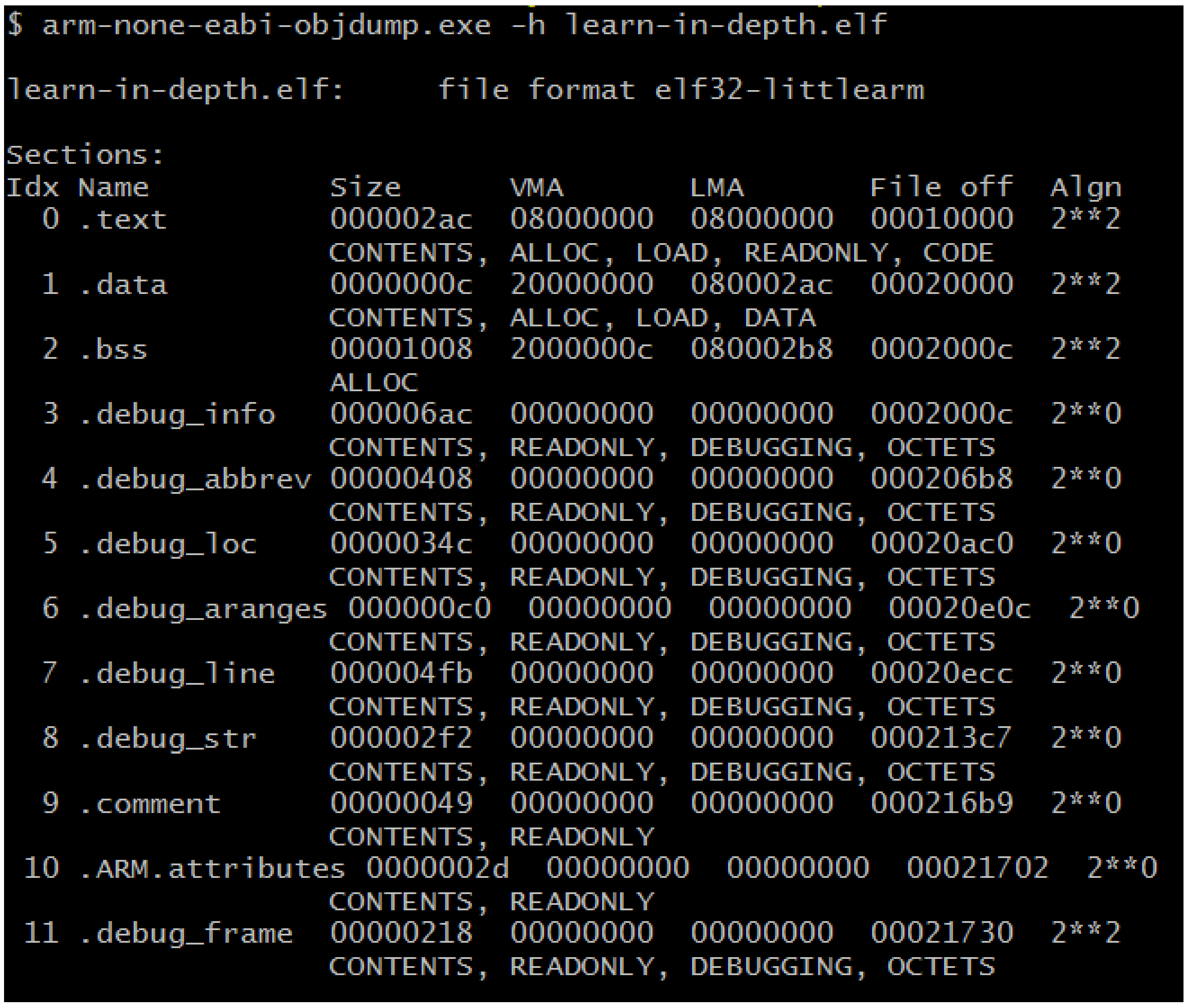
```
$ arm-none-eabi-objdump.exe -h startup.o

startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000090  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000c4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000c4  2**0
                  ALLOC
  3 .vectors      0000001c  00000000  00000000  000000c4  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  4 .debug_info   00000193  00000000  00000000  000000e0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  5 .debug_abbrev 000000e6  00000000  00000000  00000273  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  6 .debug_loc    0000007c  00000000  00000000  00000359  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  7 .debug_aranges 00000020  00000000  00000000  000003d5  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  8 .debug_line   0000018a  00000000  00000000  000003f5  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  9 .debug_str    000001ba  00000000  00000000  0000057f  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 10 .comment      0000004a  00000000  00000000  00000739  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000050  00000000  00000000  00000784  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 12 .ARM.attributes 0000002d  00000000  00000000  000007d4  2**0
                  CONTENTS, READONLY
```

startup.o symbols:



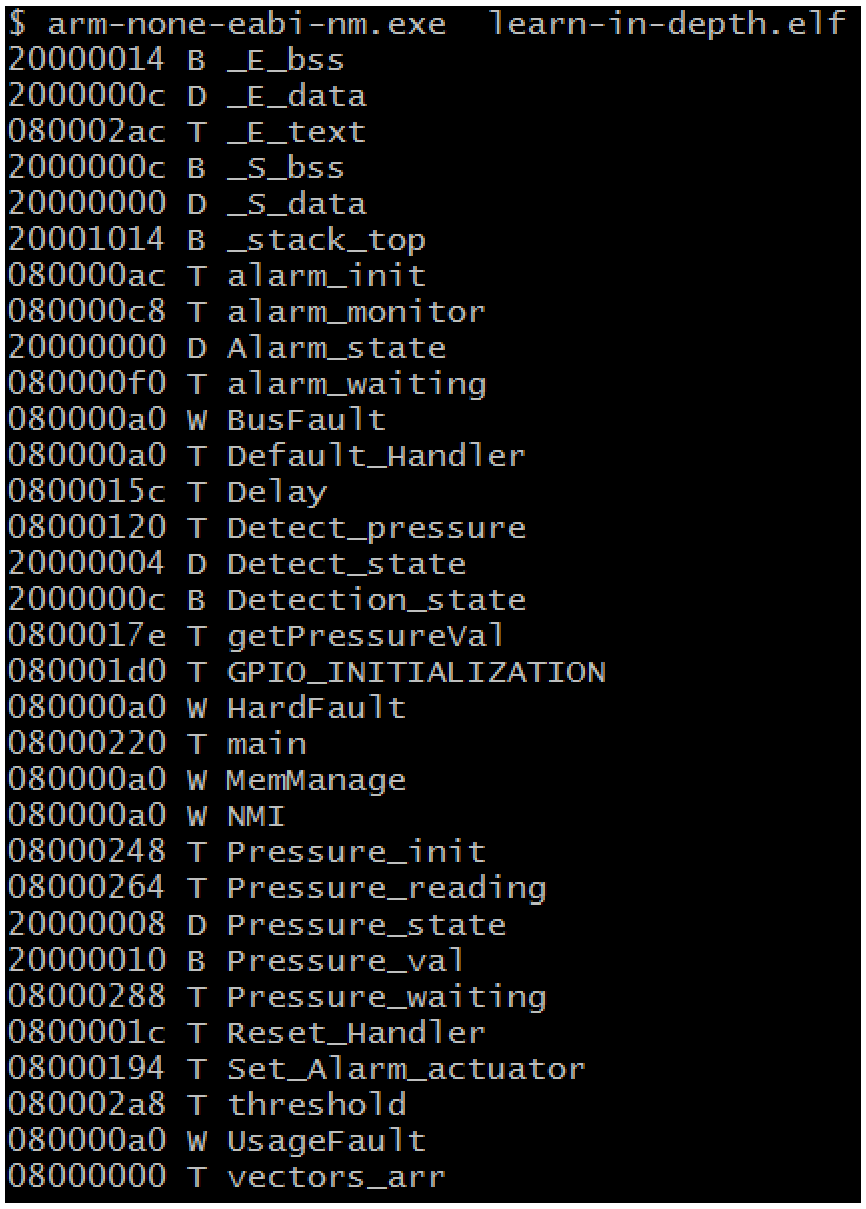
```
$ arm-none-eabi-nm.exe  startup.o
         U _E_bss
         U _E_data
         U _E_text
         U _S_bss
         U _S_data
         U _stack_top
00000084 W BusFault
00000084 T Default_Handler
00000084 W HardFault
         U main
00000084 W MemManage
00000084 W NMI
00000000 T Reset_Handler
00000084 W UsageFault
00000000 D vectors_arr
```

learn_in_depth.elf sections:



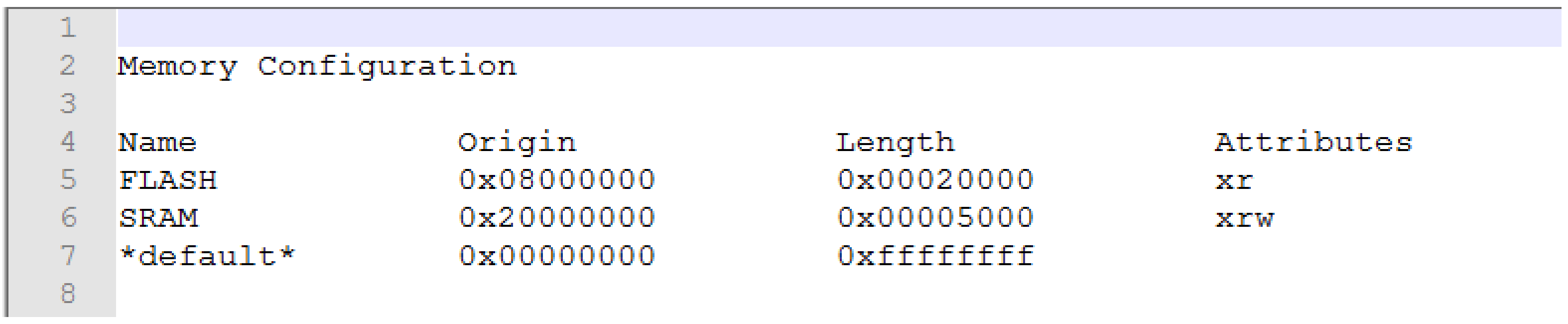
```
$ arm-none-eabi-objdump.exe -h learn-in-depth.elf

learn-in-depth.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000002ac  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         0000000c  20000000  080002ac  00020000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001008  2000000c  080002b8  0002000c  2**2
                  ALLOC
  3 .debug_info   000006ac  00000000  00000000  0002000c  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  4 .debug_abbrev 00000408  00000000  00000000  000206b8  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  5 .debug_loc    0000034c  00000000  00000000  00020ac0  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  6 .debug_aranges 000000c0  00000000  00000000  00020e0c  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  7 .debug_line   000004fb  00000000  00000000  00020ecc  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  8 .debug_str    000002f2  00000000  00000000  000213c7  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  9 .comment      00000049  00000000  00000000  000216b9  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 0000002d  00000000  00000000  00021702  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000218  00000000  00000000  00021730  2**2
                  CONTENTS, READONLY, DEBUGGING, OCTETS
```

learn_in_depth.elf symbols:

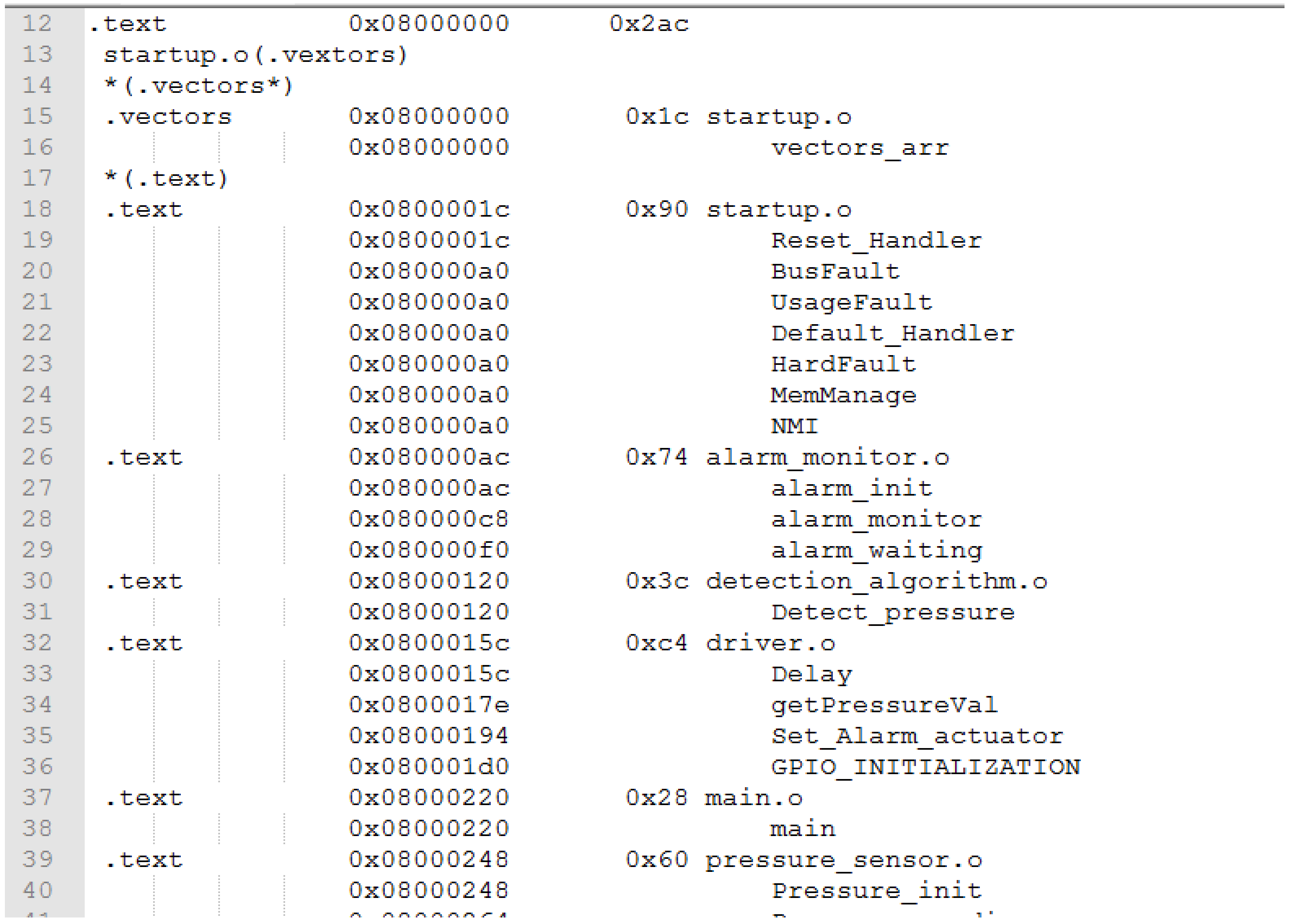


```
$ arm-none-eabi-nm.exe  learn-in-depth.elf
20000014 B _E_bss
2000000c D _E_data
080002ac T _E_text
2000000c B _S_bss
20000000 D _S_data
20001014 B _stack_top
080000ac T alarm_init
080000c8 T alarm_monitor
20000000 D Alarm_state
080000f0 T alarm_waiting
080000a0 W BusFault
080000a0 T Default_Handler
0800015c T Delay
08000120 T Detect_pressure
20000004 D Detect_state
2000000c B Detection_state
0800017e T getPressureVal
080001d0 T GPIO_INITIALIZATION
080000a0 W HardFault
08000220 T main
080000a0 W MemManage
080000a0 W NMI
08000248 T Pressure_init
08000264 T Pressure_reading
20000008 D Pressure_state
20000010 B Pressure_val
08000288 T Pressure_waiting
0800001c T Reset_Handler
08000194 T Set_Alarm_actuator
080002a8 T threshold
080000a0 W UsageFault
08000000 T vectors_arr
```

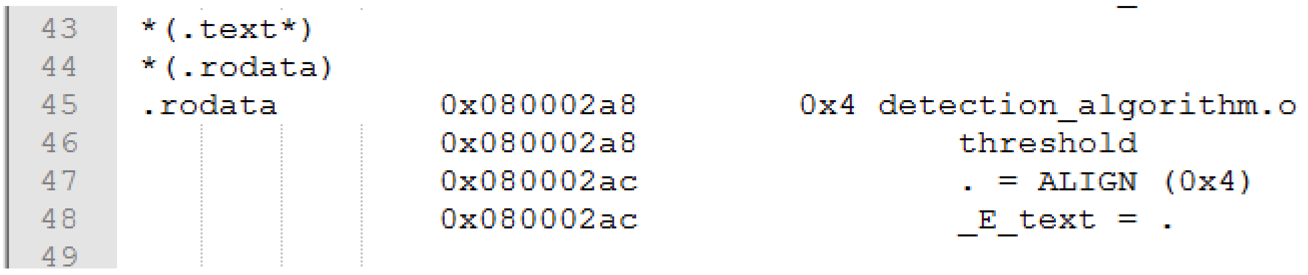
Map_file.map:



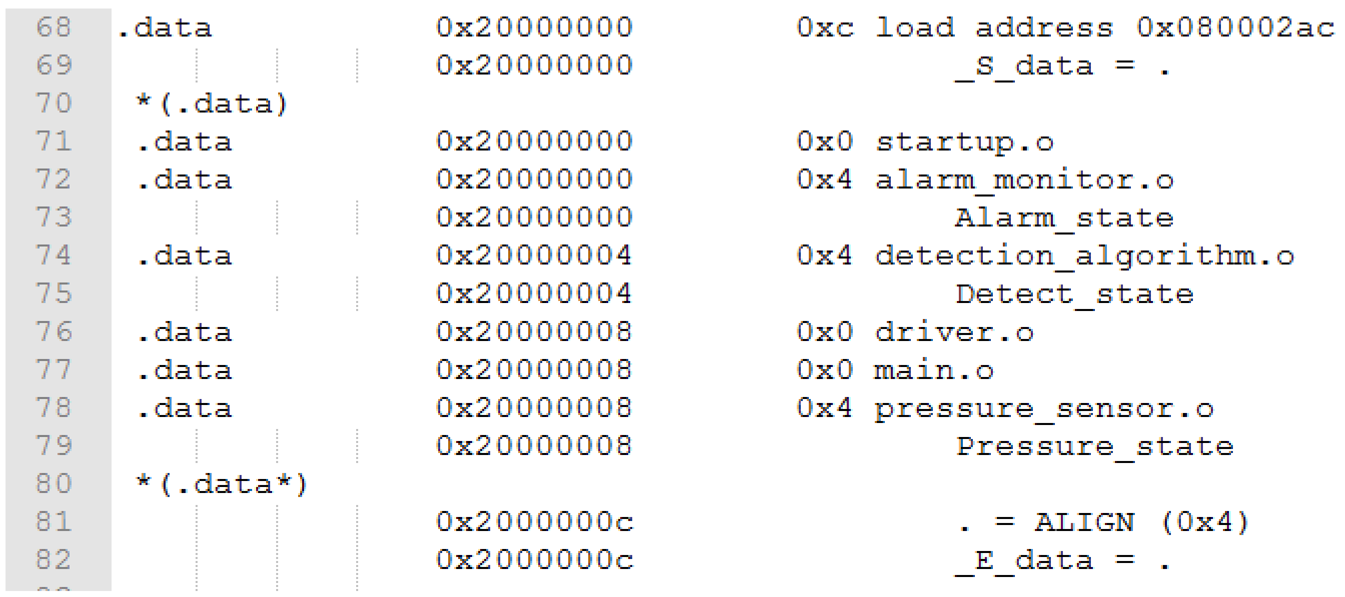
```
Memory Configuration

Name             Origin             Length             Attributes
FLASH            0x08000000         0x00020000         xr
SRAM             0x20000000         0x00005000         xrw
*default*        0x00000000         0xffffffff
```



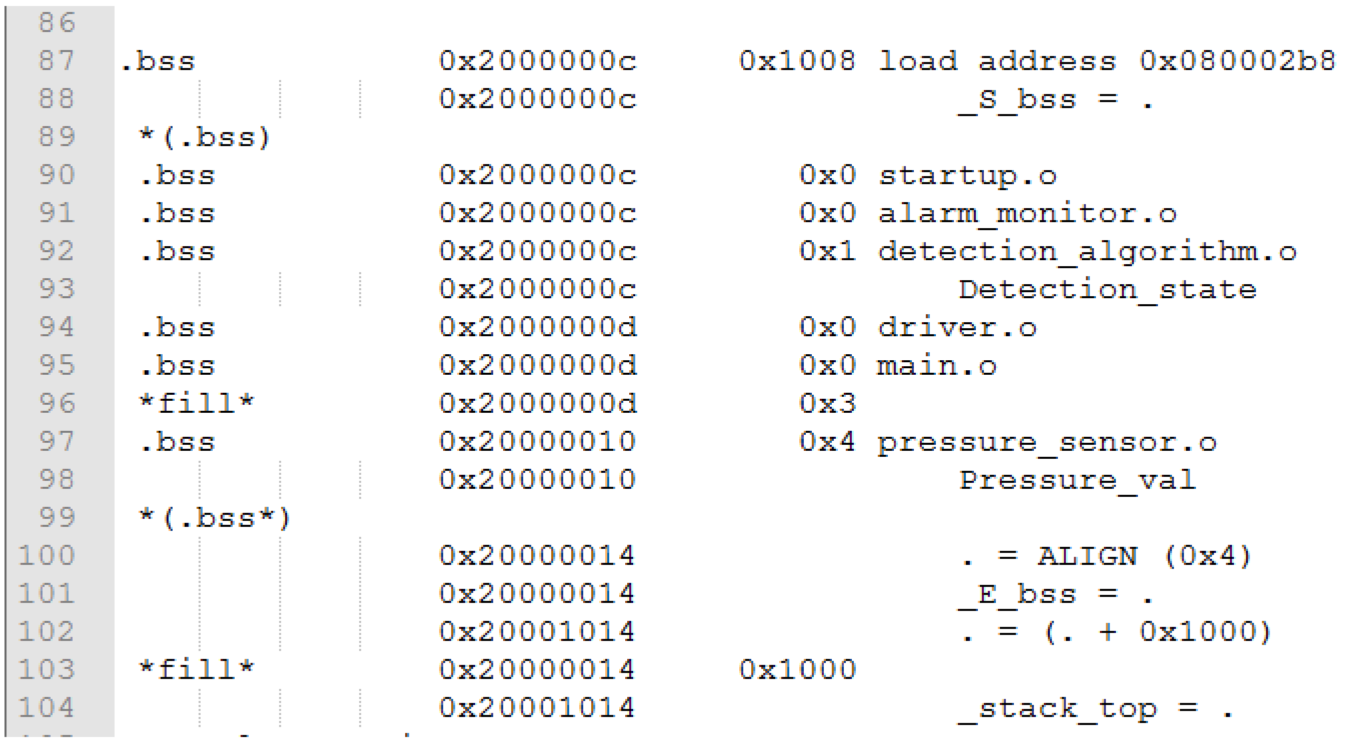
```
.text           0x08000000      0x2ac
 startup.o(.vextors)
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors_arr
 *(.text)
 .text          0x0800001c       0x90 startup.o
                0x0800001c                Reset_Handler
                0x080000a0                BusFault
                0x080000a0                UsageFault
                0x080000a0                Default_Handler
                0x080000a0                HardFault
                0x080000a0                MemManage
                0x080000a0                NMI
 .text          0x080000ac       0x74 alarm_monitor.o
                0x080000ac                alarm_init
                0x080000c8                alarm_monitor
                0x080000f0                alarm_waiting
 .text          0x08000120       0x3c detection_algorithm.o
                0x08000120                Detect_pressure
 .text          0x0800015c       0xc4 driver.o
                0x0800015c                Delay
                0x0800017e                getPressureVal
                0x08000194                Set_Alarm_actuator
                0x080001d0                GPIO_INITIALIZATION
 .text          0x08000220       0x28 main.o
                0x08000220                main
 .text          0x08000248       0x60 pressure_sensor.o
                0x08000248                Pressure_init
```



```
 *(.text*)
 *(.rodata)
 .rodata        0x080002a8        0x4 detection_algorithm.o
                0x080002a8                threshold
                0x080002ac                . = ALIGN (0x4)
                0x080002ac                _E_text = .
```



```
.data           0x20000000        0xc load address 0x080002ac
                0x20000000                _S_data = .
 *(.data)
 .data          0x20000000        0x0 startup.o
 .data          0x20000000        0x4 alarm_monitor.o
                0x20000000                Alarm_state
 .data          0x20000004        0x4 detection_algorithm.o
                0x20000004                Detect_state
 .data          0x20000008        0x0 driver.o
 .data          0x20000008        0x0 main.o
 .data          0x20000008        0x4 pressure_sensor.o
                0x20000008                Pressure_state
 *(.data*)
                0x2000000c                . = ALIGN (0x4)
                0x2000000c                _E_data = .
```



```

.bss            0x2000000c      0x1008 load address 0x080002b8
                0x2000000c              _S_bss = .
 *(.bss)
 .bss           0x2000000c         0x0 startup.o
 .bss           0x2000000c         0x0 alarm_monitor.o
 .bss           0x2000000c         0x1 detection_algorithm.o
                0x2000000c              Detection_state
 .bss           0x2000000d         0x0 driver.o
 .bss           0x2000000d         0x0 main.o
 *fill*         0x2000000d         0x3
 .bss           0x20000010         0x4 pressure_sensor.o
                0x20000010              Pressure_val
 *(.bss*)
                0x20000014              . = ALIGN (0x4)
                0x20000014              _E_bss = .
                0x20001014              . = (. + 0x1000)
 *fill*         0x20000014      0x1000
                0x20001014              _stack_top = .
```