Mastering Embedded System Online Diploma

www.learn-in-depth.com

First Term (Final Project 1)

Eng. Bassam Khamis Mansour

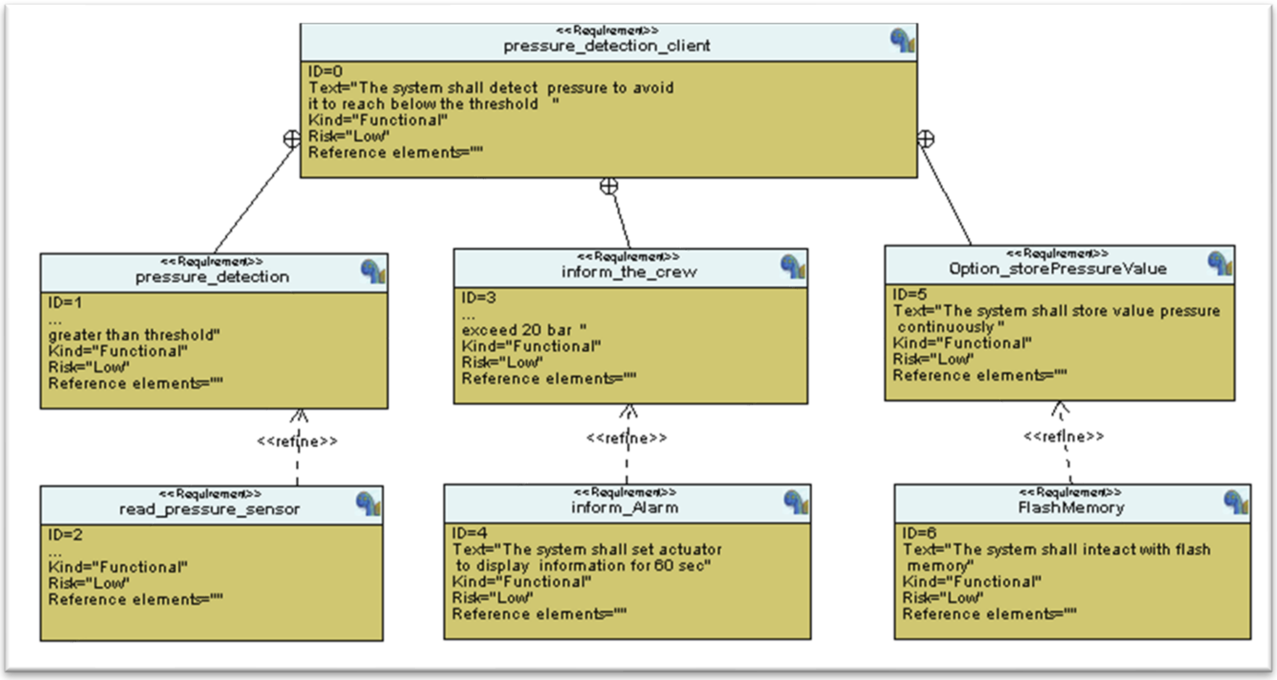
My profile:

bassammansour96@gmail.com (learn-in-depth.com)

Intro:

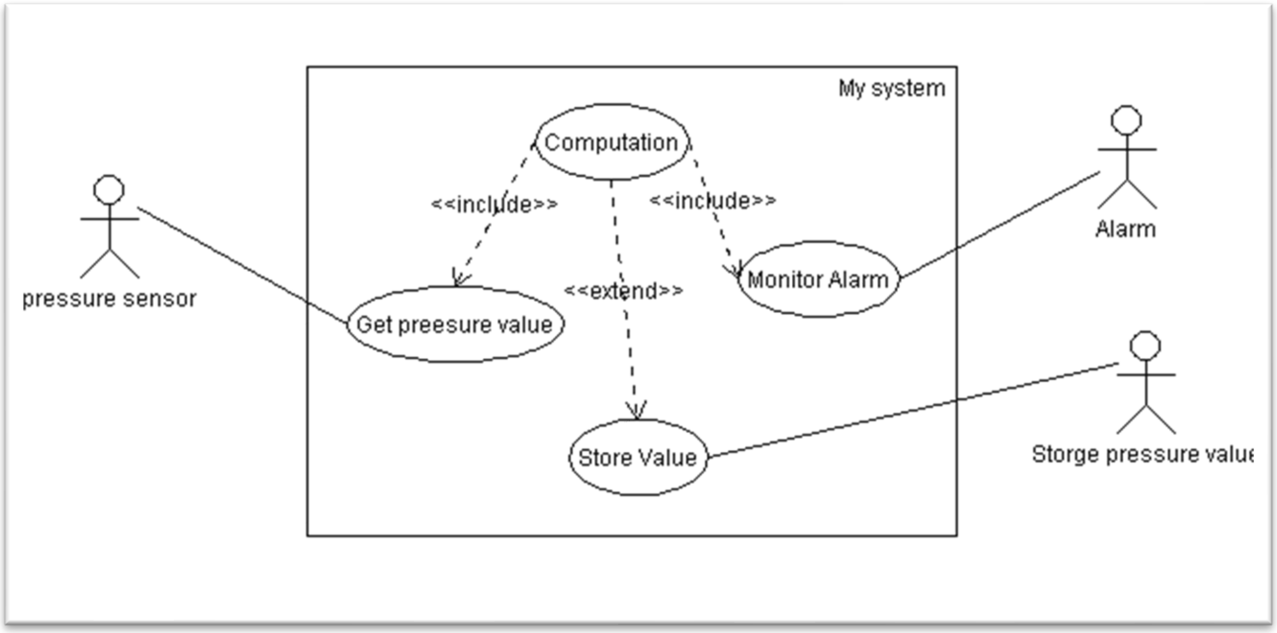
Required software to control the robot such that it keeps moving until faced an object, when it as distance 50 cm or less, the robot at this moment the robot must be stop.

Requirement's diagram:

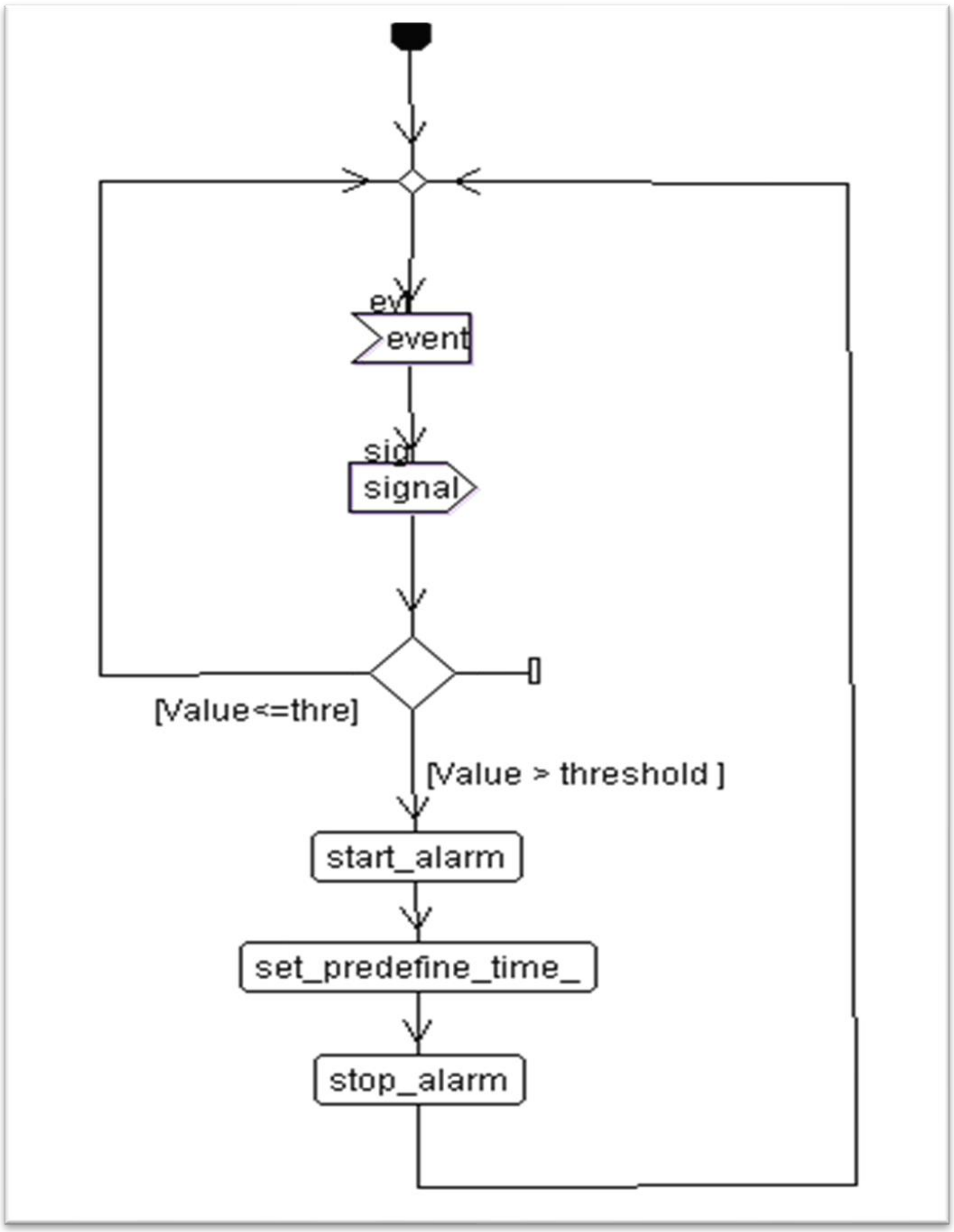


System Analysis:

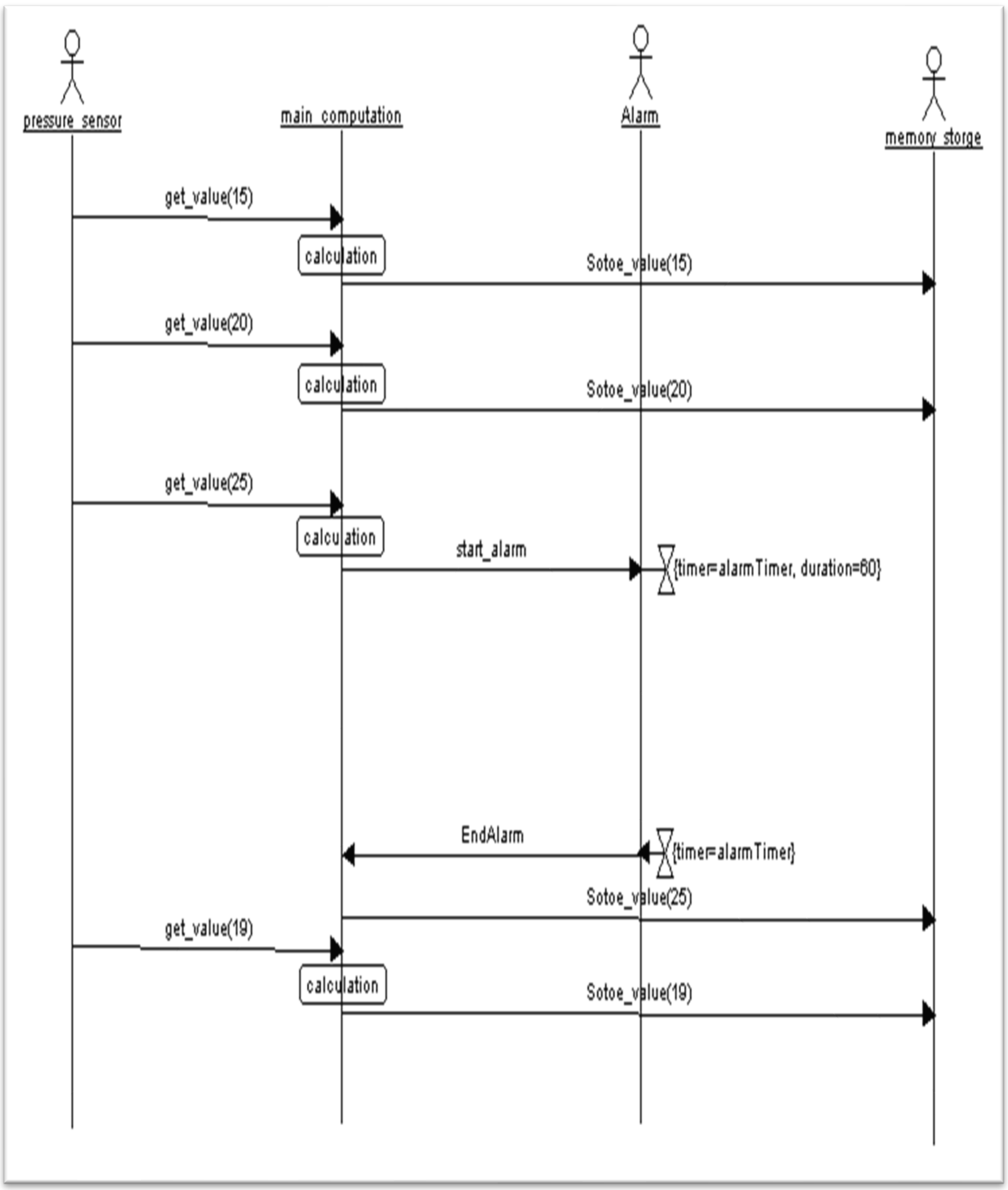
- **Use case diagram:**



- Activity diagram:

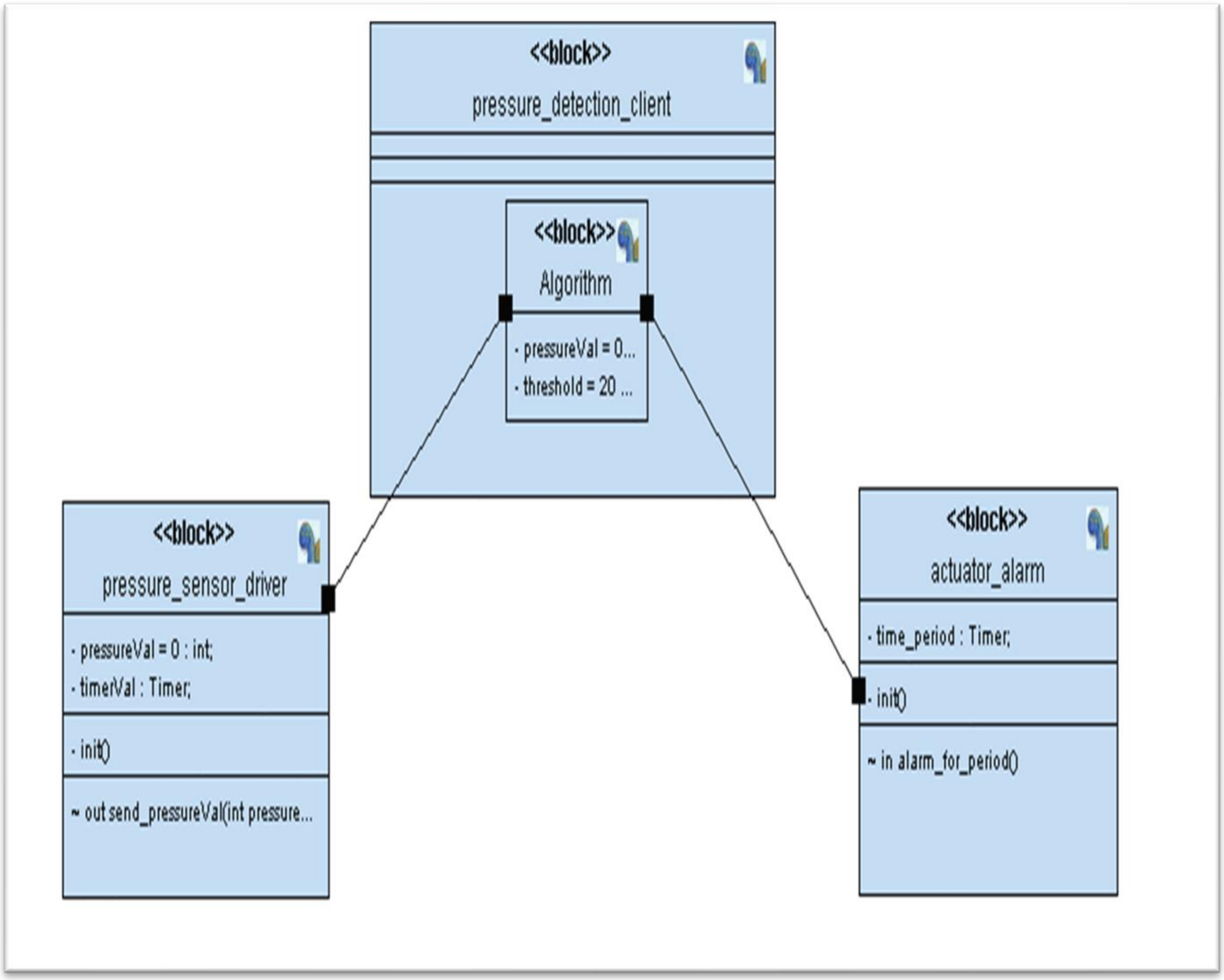


Sequence diagram:

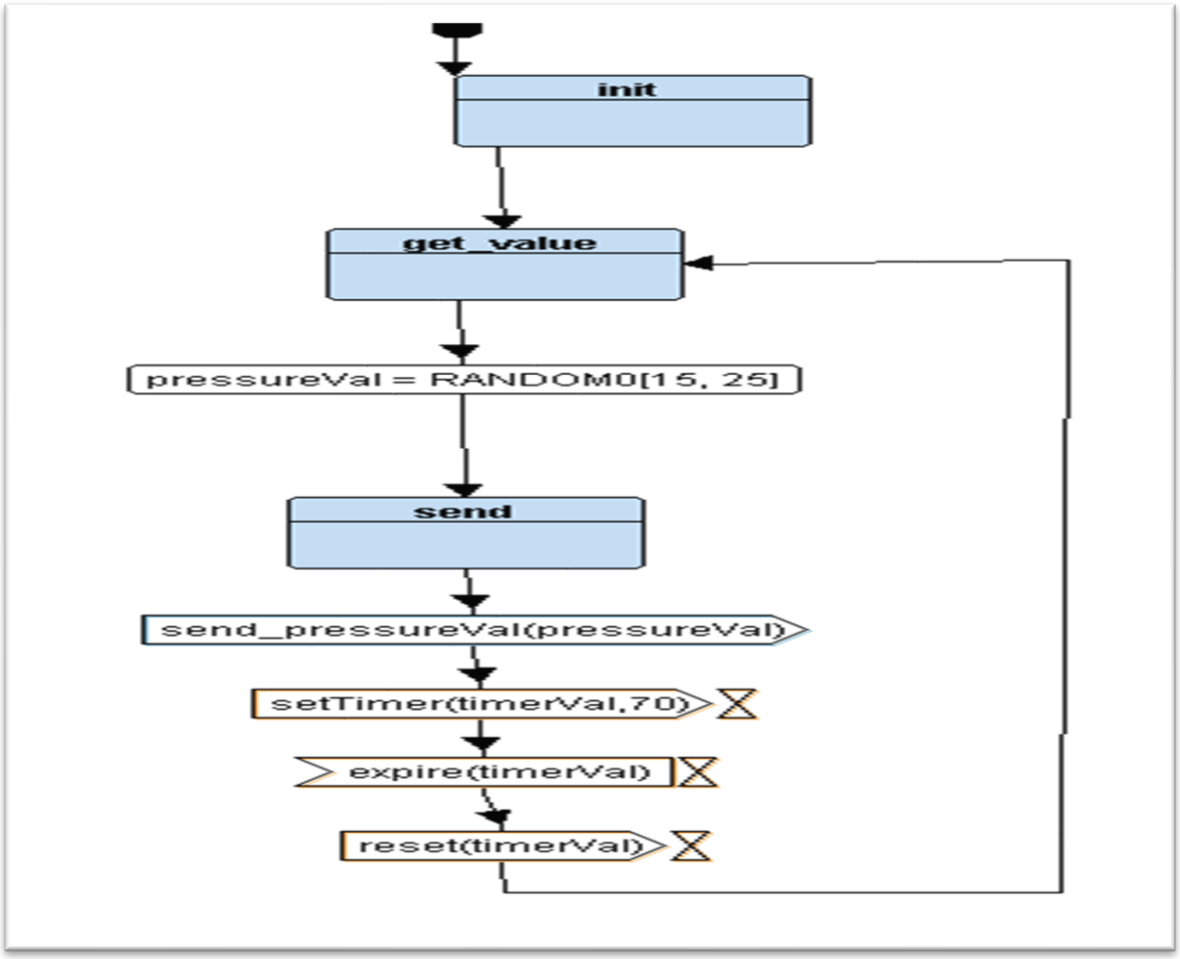


System design:

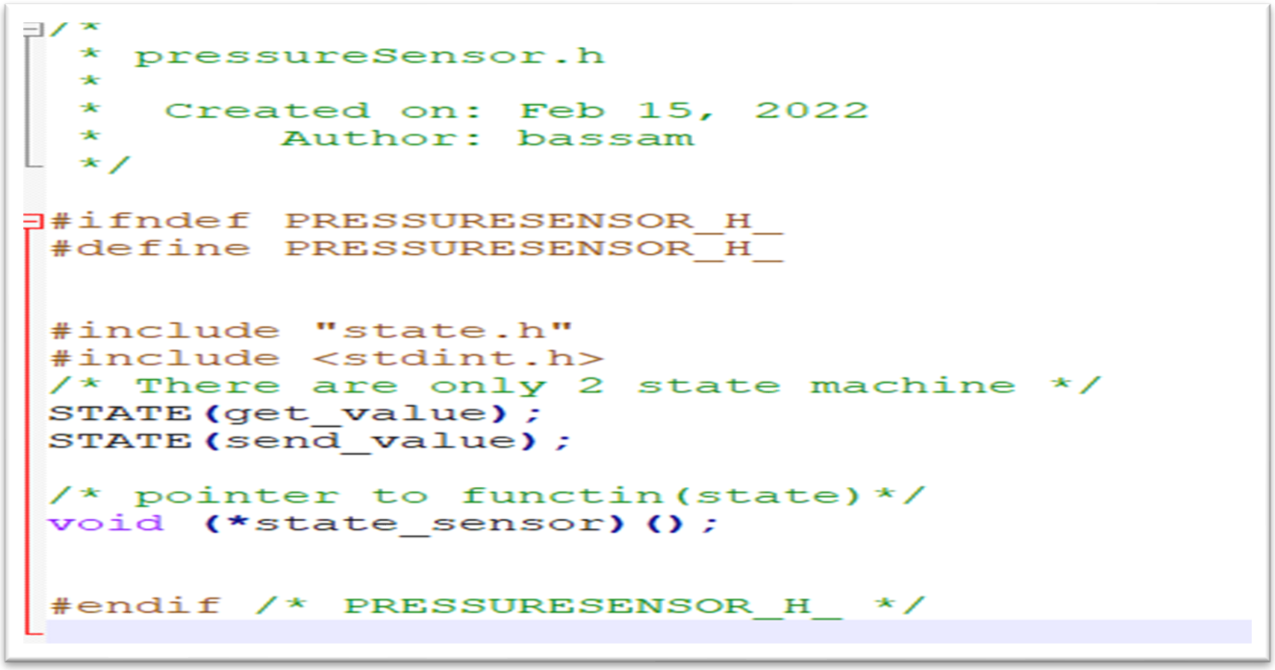
- Main block diagram:



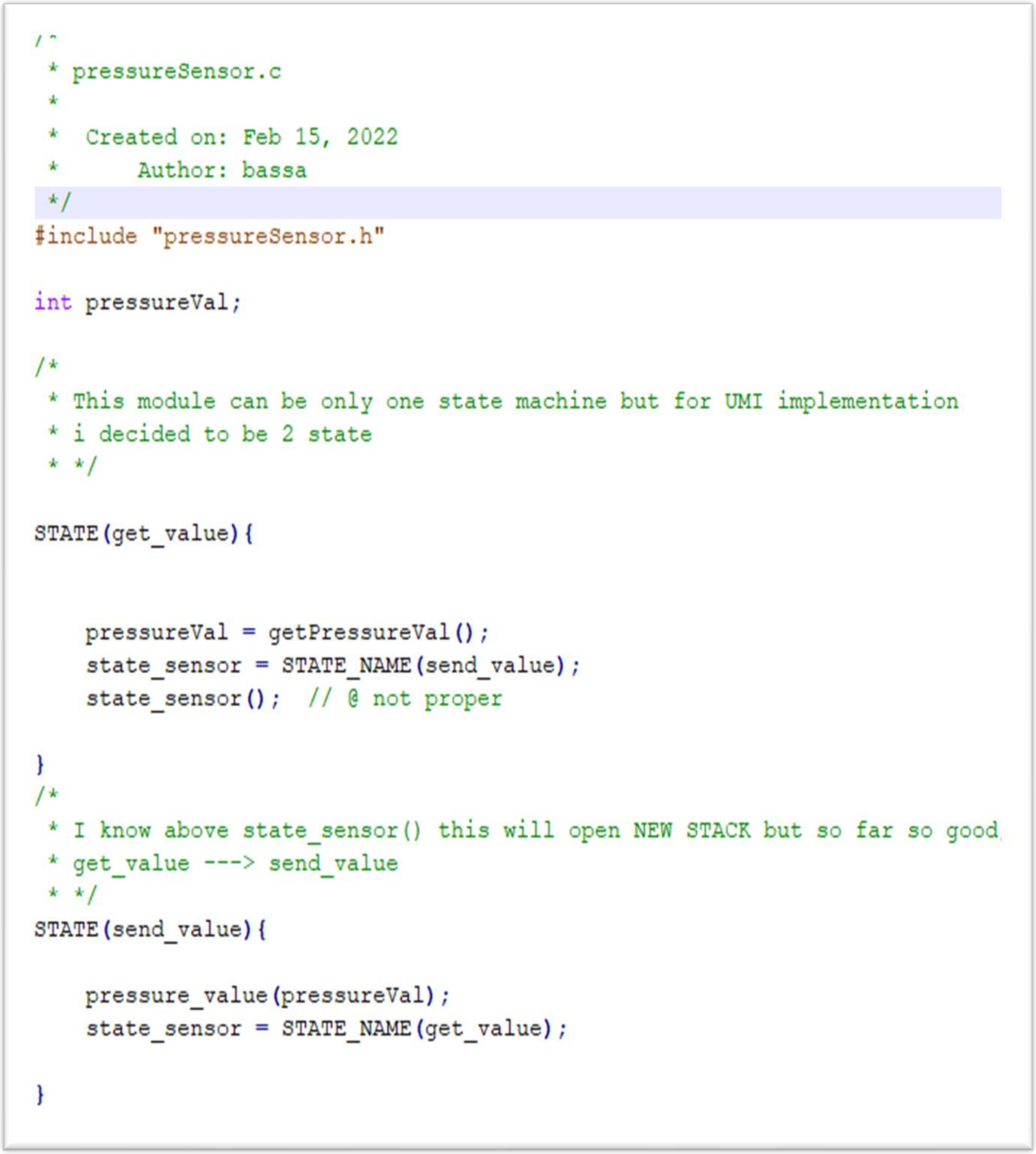
- **Pressure sensor state machine:**



- **Pressure sensor module implementation:**

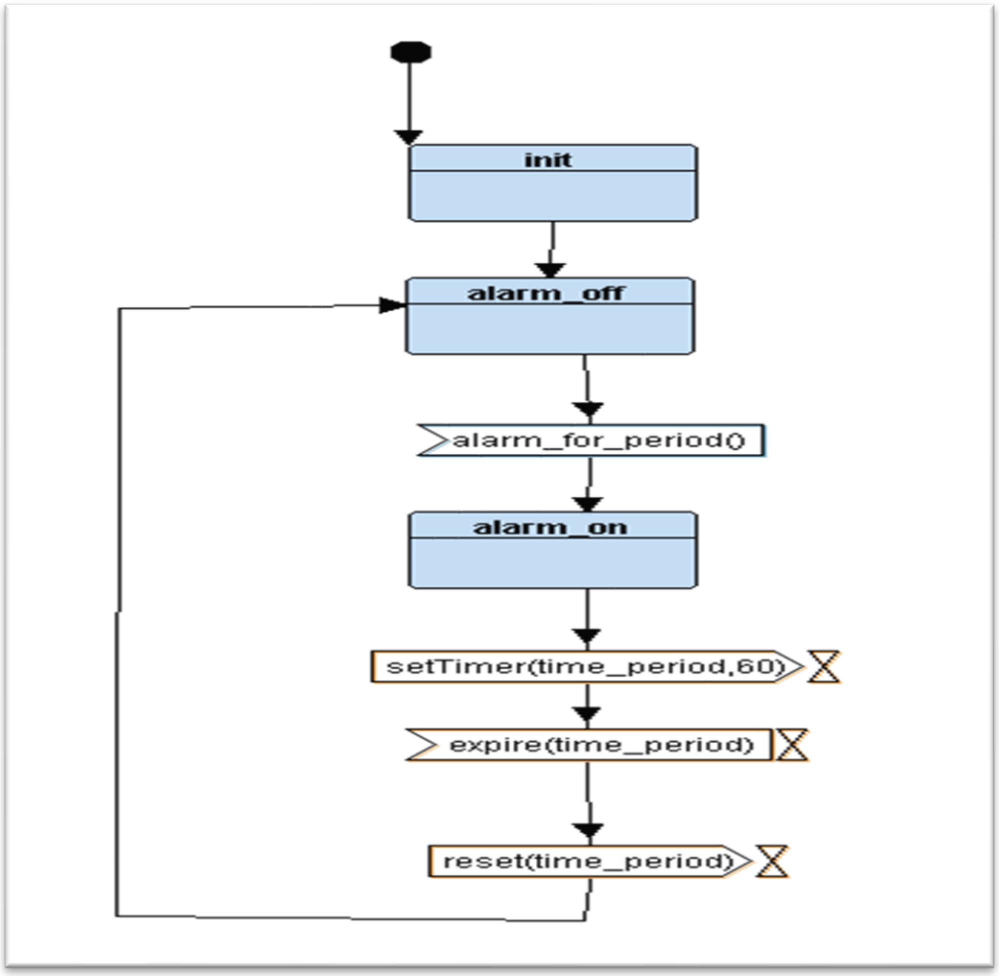


```
/*
 * pressureSensor.h
 *
 *  Created on: Feb 15, 2022
 *      Author: bassam
 */

#ifndef PRESSURESENSOR_H_
#define PRESSURESENSOR_H_

#include "state.h"
#include <stdint.h>
/* There are only 2 state machine */
STATE(get_value);
STATE(send_value);

/* pointer to functin(state)*/
void (*state_sensor)();


#endif /* PRESSURESENSOR_H_ */
```

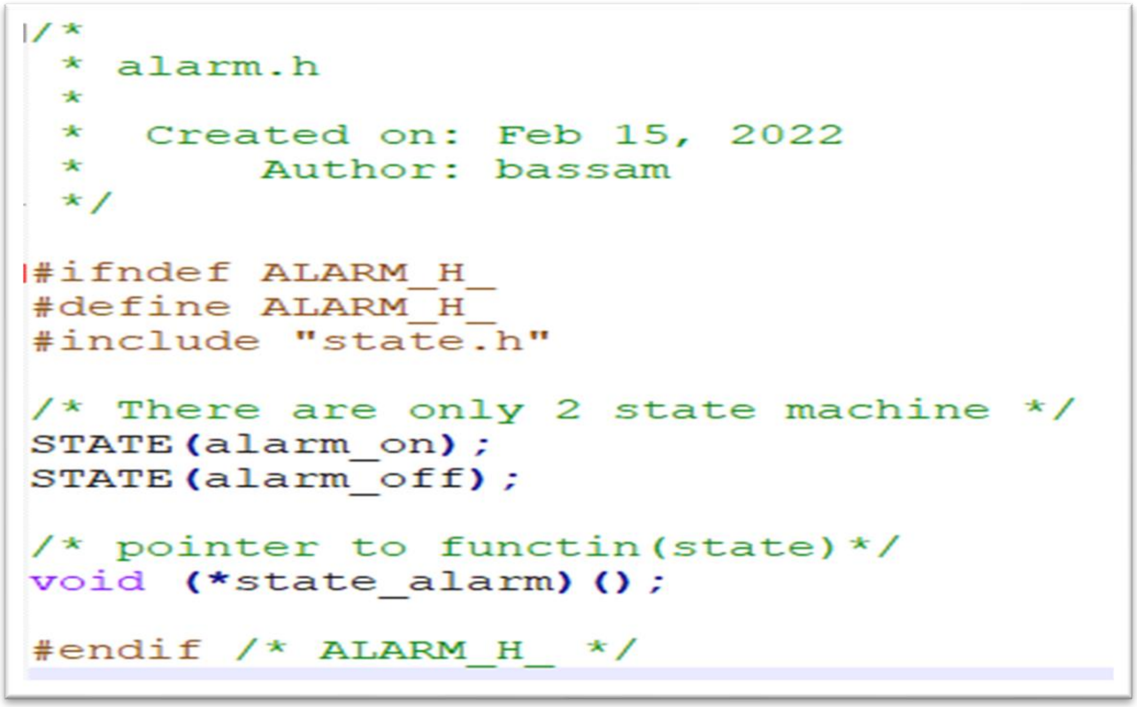


```
/*
 * pressureSensor.c
 *
 *  Created on: Feb 15, 2022
 *      Author: bassa
 */
#include "pressureSensor.h"

int pressureVal;

/*
 * This module can be only one state machine but for UMI implementation
 * i decided to be 2 state
 * */

STATE(get_value){


    pressureVal = getPressureVal();
    state_sensor = STATE_NAME(send_value);
    state_sensor();  // @ not proper

}
/*
 * I know above state_sensor() this will open NEW STACK but so far so good
 * get_value ---> send_value
 * */
STATE(send_value){

    pressure_value(pressureVal);
    state_sensor = STATE_NAME(get_value);

}
```

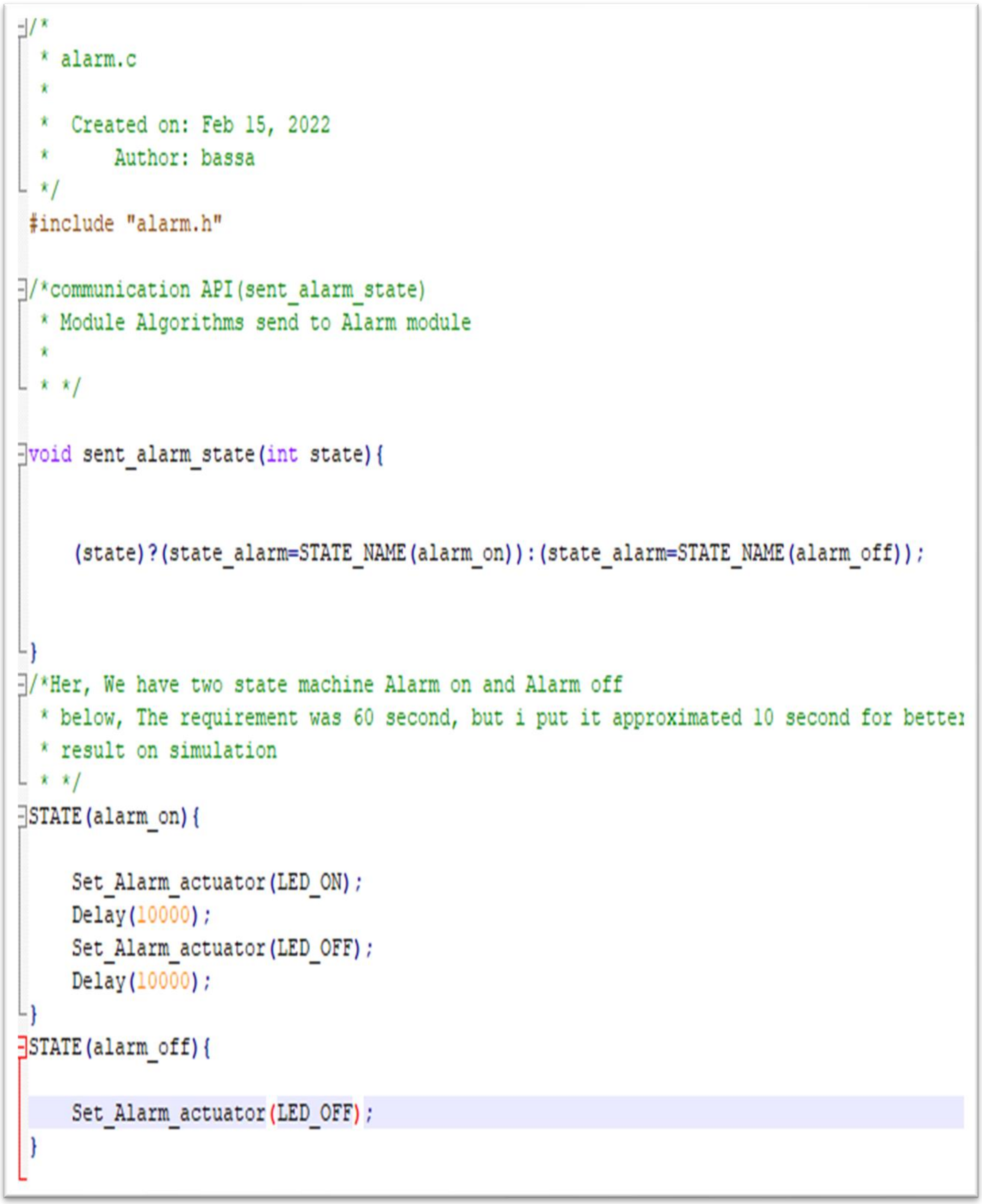
- Alarm actuator state machine:



- Alarm implementation:

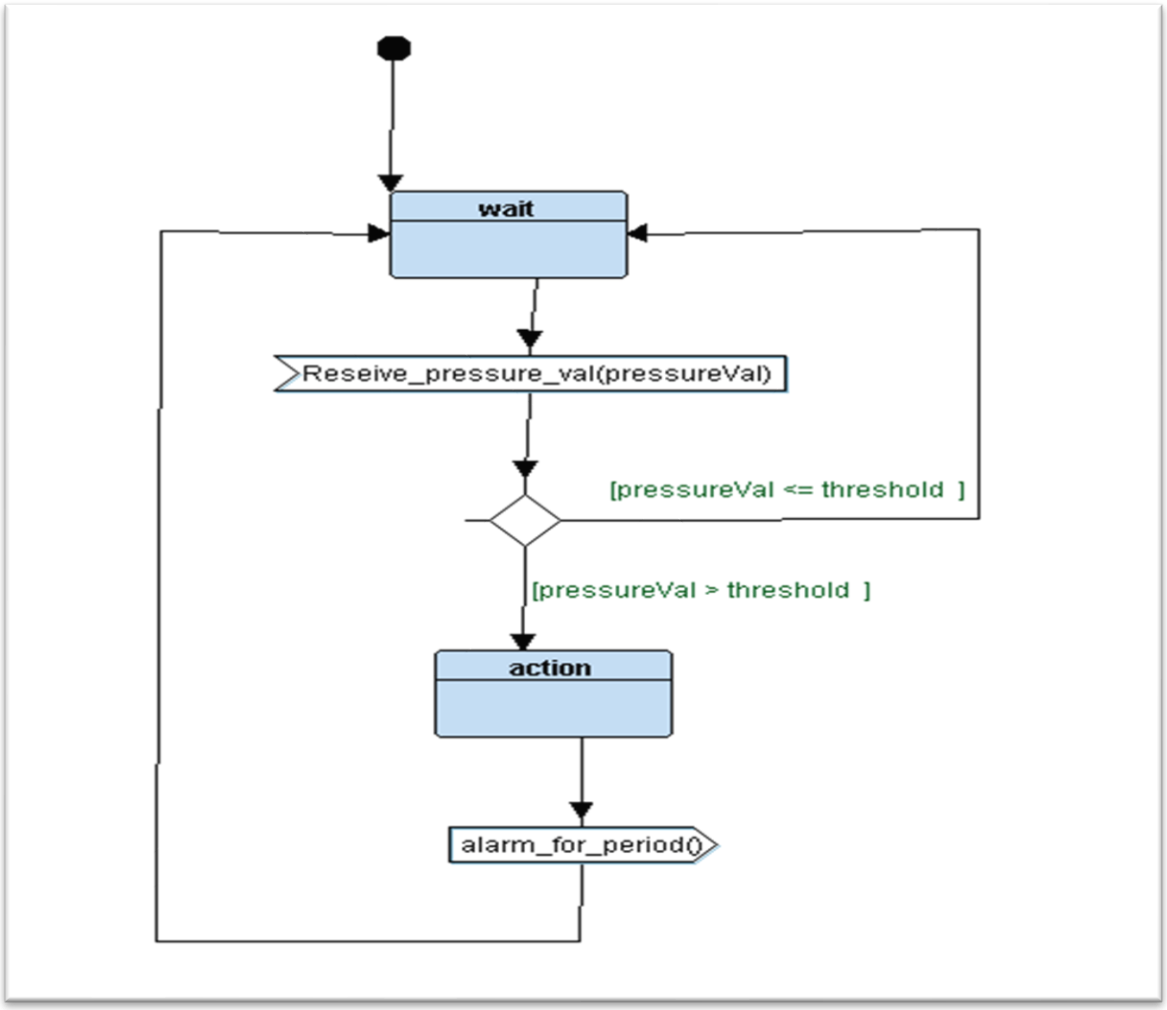


```
/*
 *  alarm.h
 *
 *   Created on: Feb 15, 2022
 *       Author: bassam
 */

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

/* There are only 2 state machine */
STATE(alarm_on);
STATE(alarm_off);

/* pointer to functin(state)*/
void (*state_alarm)();

#endif /* ALARM_H_ */
```

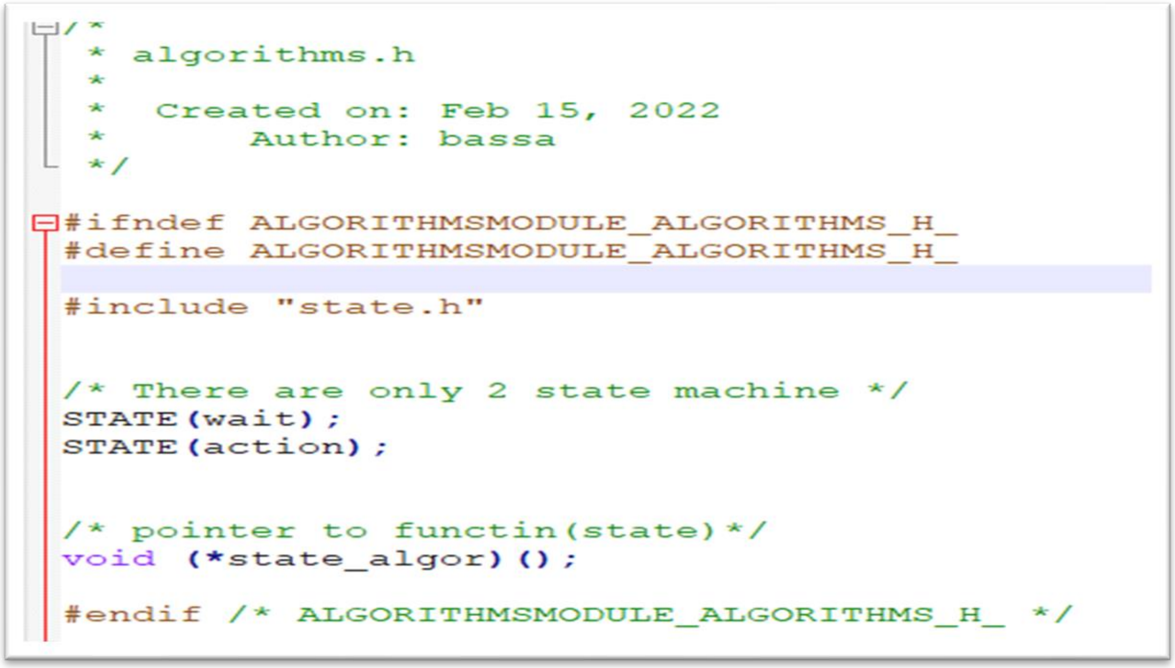


```c
/*
 * alarm.c
 *
 *  Created on: Feb 15, 2022
 *      Author: bassa
 */
#include "alarm.h"

/*communication API(sent_alarm_state)
 * Module Algorithms send to Alarm module
 *
 * */

void sent_alarm_state(int state){


    (state)?(state_alarm=STATE_NAME(alarm_on)):(state_alarm=STATE_NAME(alarm_off));



}
/*Her, We have two state machine Alarm on and Alarm off
 * below, The requirement was 60 second, but i put it approximated 10 second for bette
 * result on simulation
 * */
STATE(alarm_on){

    Set_Alarm_actuator(LED_ON);
    Delay(10000);
    Set_Alarm_actuator(LED_OFF);
    Delay(10000);
}
STATE(alarm_off){

    Set_Alarm_actuator(LED_OFF);
}
```

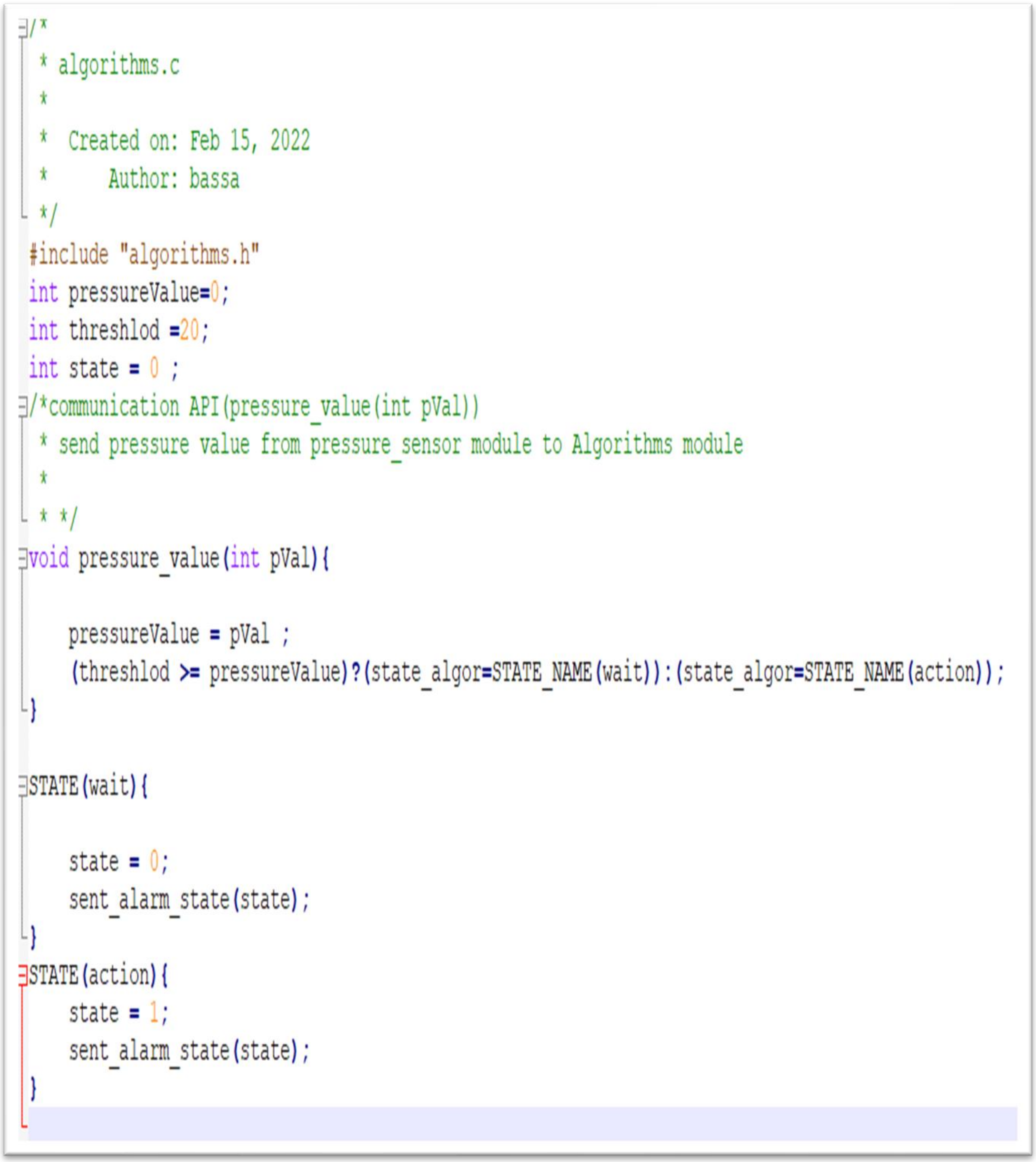
- Algorithms state machine:


wait
Reseive_pressure_val(pressureVal)
[pressureVal <= threshold]
[pressureVal > threshold]
action
alarm_for_period()

- algorithms implementation:

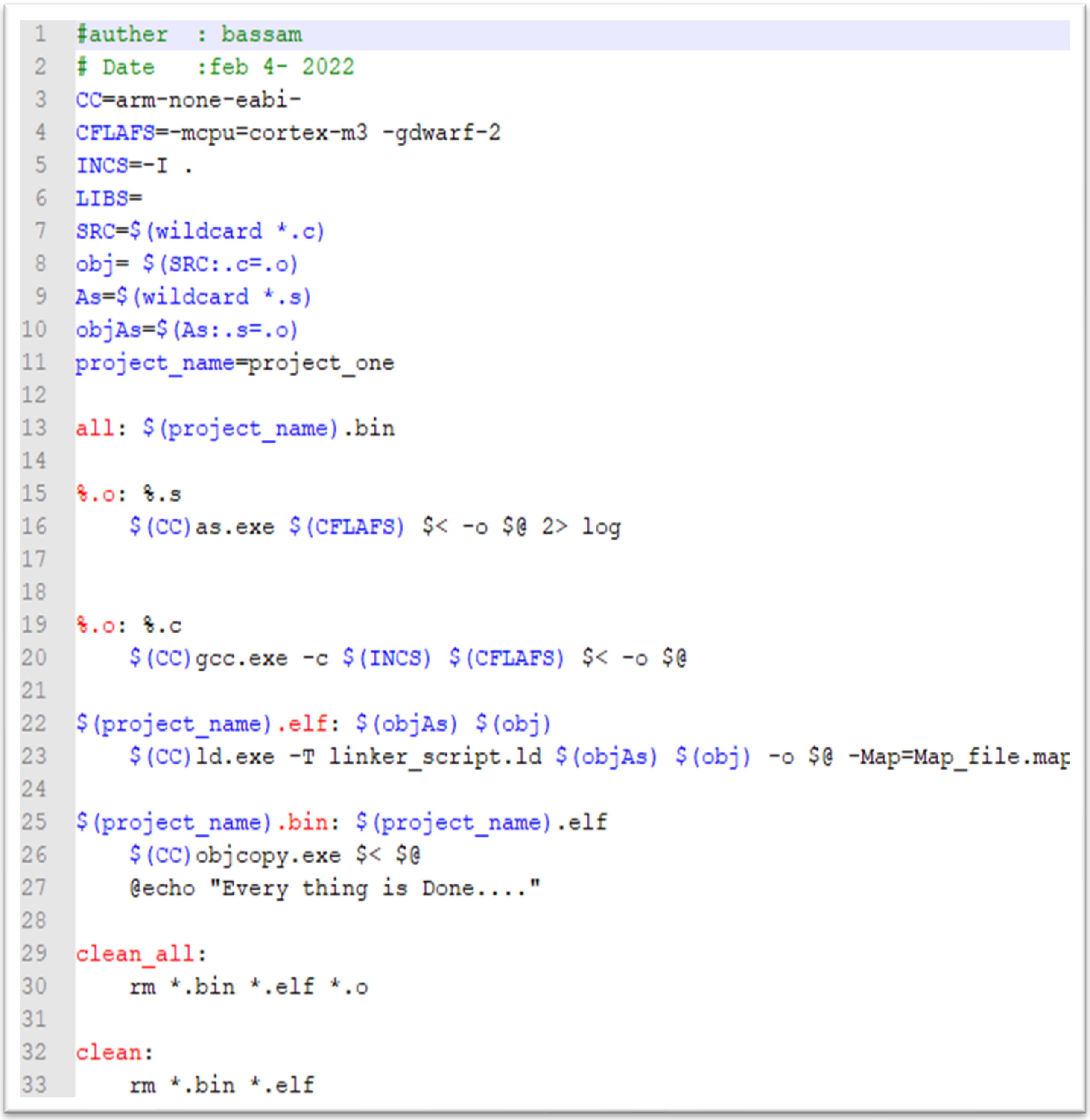


```
/*
 * algorithms.h
 *
 *  Created on: Feb 15, 2022
 *      Author: bassa
 */

#ifndef ALGORITHMSMODULE_ALGORITHMS_H_
#define ALGORITHMSMODULE_ALGORITHMS_H_

#include "state.h"

/* There are only 2 state machine */
STATE(wait);
STATE(action);

/* pointer to functin(state)*/
void (*state_algor)();

#endif /* ALGORITHMSMODULE_ALGORITHMS_H_ */
```



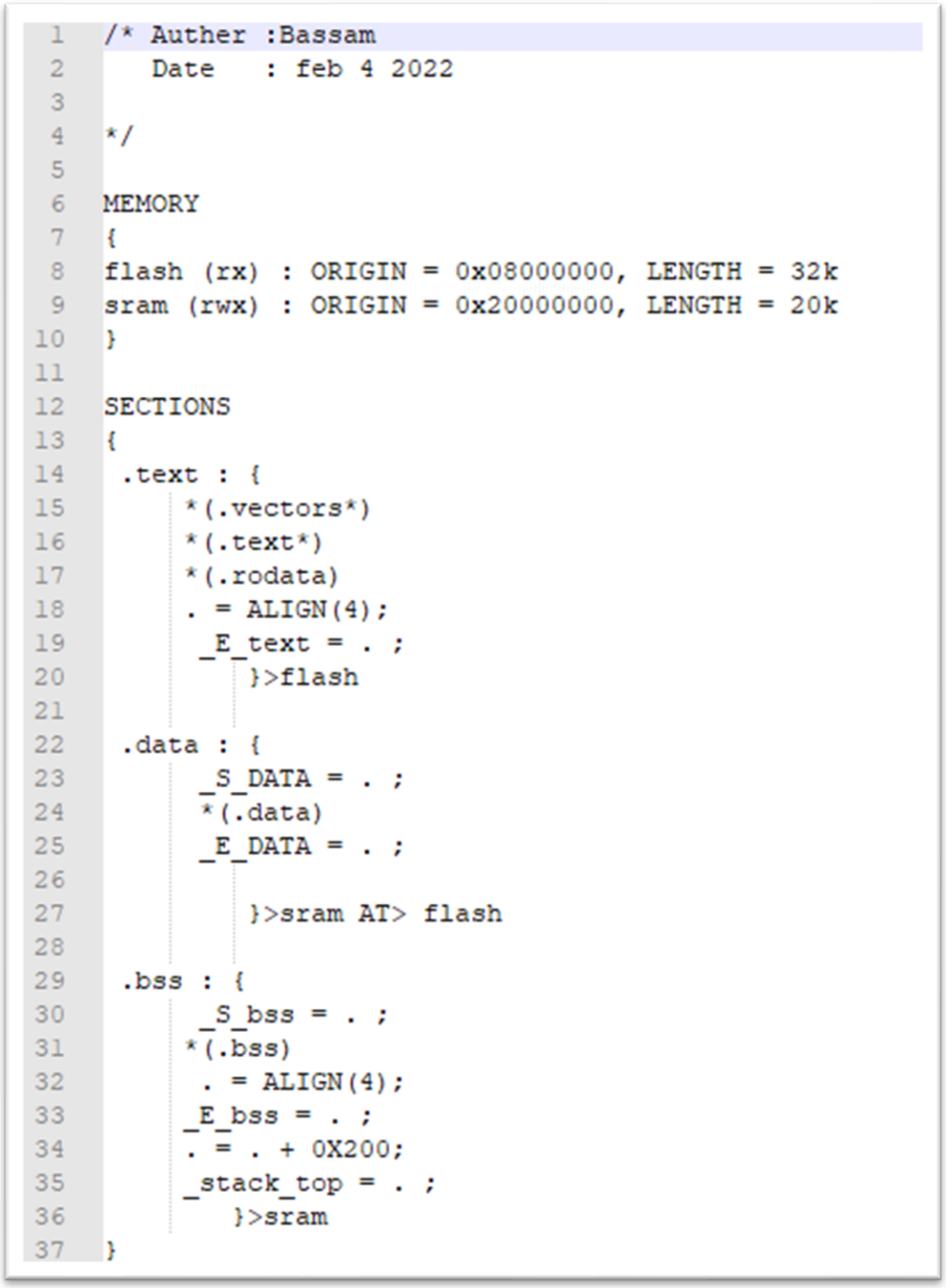
```c
/*
 * algorithms.c
 *
 *  Created on: Feb 15, 2022
 *      Author: bassa
 */
#include "algorithms.h"
int pressureValue=0;
int threshlod =20;
int state = 0 ;
/*communication API(pressure_value(int pVal))
 * send pressure value from pressure_sensor module to Algorithms module
 *
 * */
void pressure_value(int pVal){

    pressureValue = pVal ;
    (threshlod >= pressureValue)?(state_algor=STATE_NAME(wait)):(state_algor=STATE_NAME(action));
}


STATE(wait){

    state = 0;
    sent_alarm_state(state);
}
STATE(action){
    state = 1;
    sent_alarm_state(state);
}
```

Make file:



```makefile
#auther  : bassam
# Date   :feb 4- 2022
CC=arm-none-eabi-
CFLAFS=-mcpu=cortex-m3 -gdwarf-2
INCS=-I .
LIBS=
SRC=$(wildcard *.c)
obj= $(SRC:.c=.o)
As=$(wildcard *.s)
objAs=$(As:.s=.o)
project_name=project_one

all: $(project_name).bin

%.o: %.s
	$(CC)as.exe $(CFLAFS) $< -o $@ 2> log


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

$(project_name).elf: $(objAs) $(obj)
	$(CC)ld.exe -T linker_script.ld $(objAs) $(obj) -o $@ -Map=Map_file.map

$(project_name).bin: $(project_name).elf
	$(CC)objcopy.exe $< $@
	@echo "Every thing is Done...."

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

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

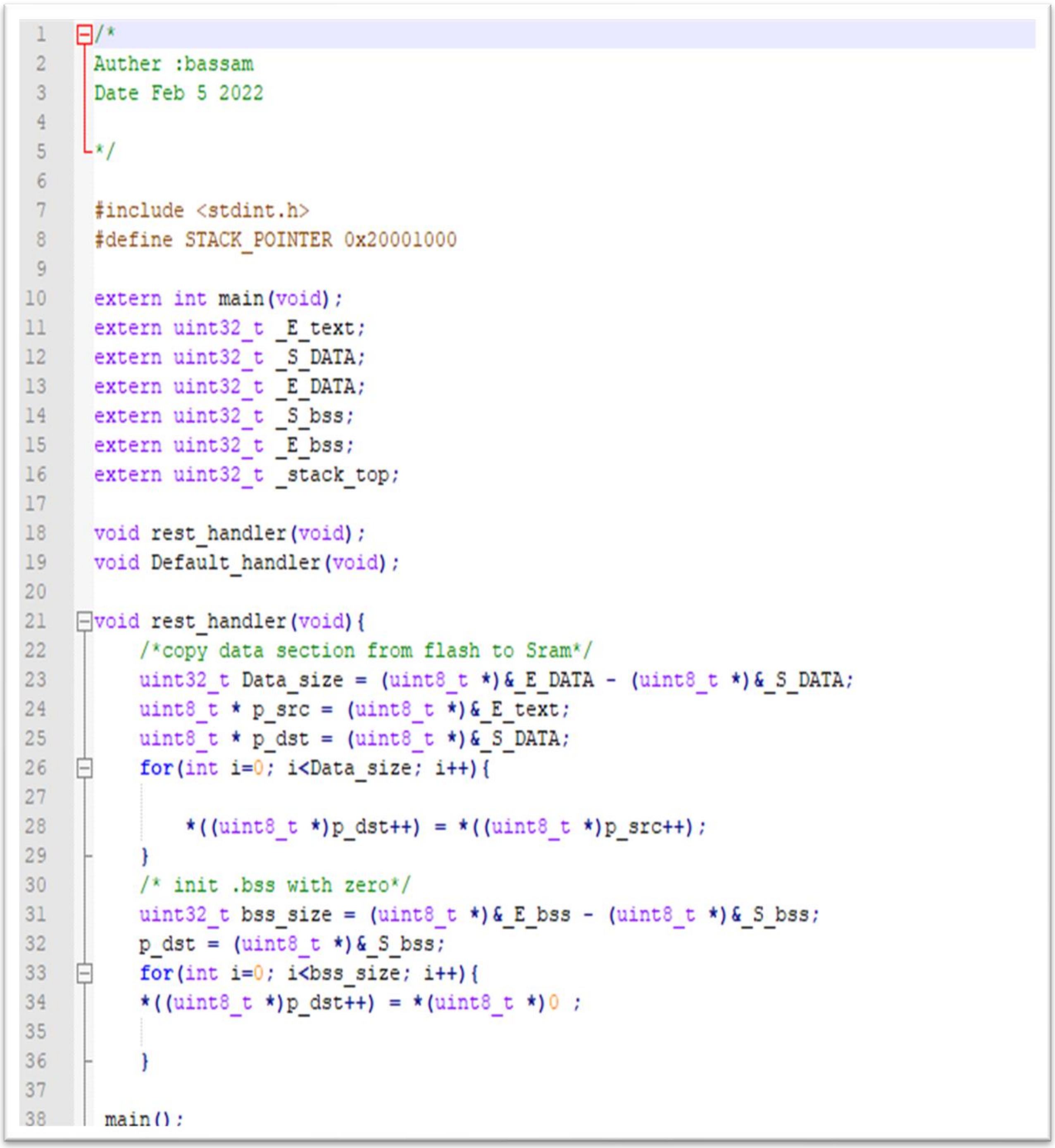
Linker script:



```
/* Auther :Bassam
   Date   : feb 4 2022

*/

MEMORY
{
flash (rx) : ORIGIN = 0x08000000, LENGTH = 32k
sram (rwx) : ORIGIN = 0x20000000, LENGTH = 20k
}

SECTIONS
{
 .text : {
      *(.vectors*)
      *(.text*)
      *(.rodata)
      . = ALIGN(4);
       _E_text = . ;
         }>flash

 .data : {
       _S_DATA = . ;
       *(.data)
       _E_DATA = . ;

         }>sram AT> flash

 .bss : {
       _S_bss = . ;
      *(.bss)
       . = ALIGN(4);
      _E_bss = . ;
      . = . + 0X200;
      _stack_top = . ;
        }>sram
}
```

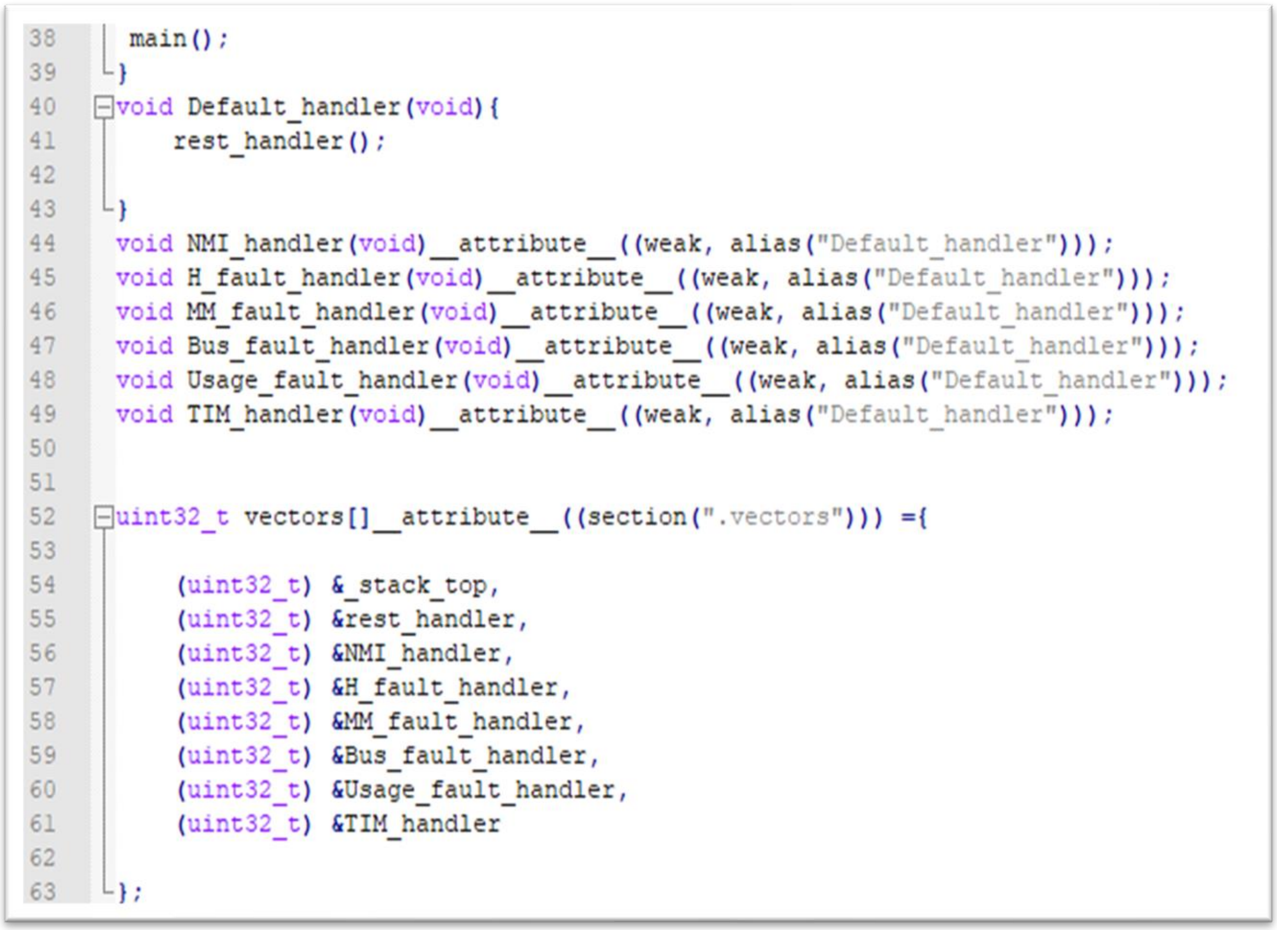
Note: above I create stack size equal 0x200(512 b)

Startup file dot c:



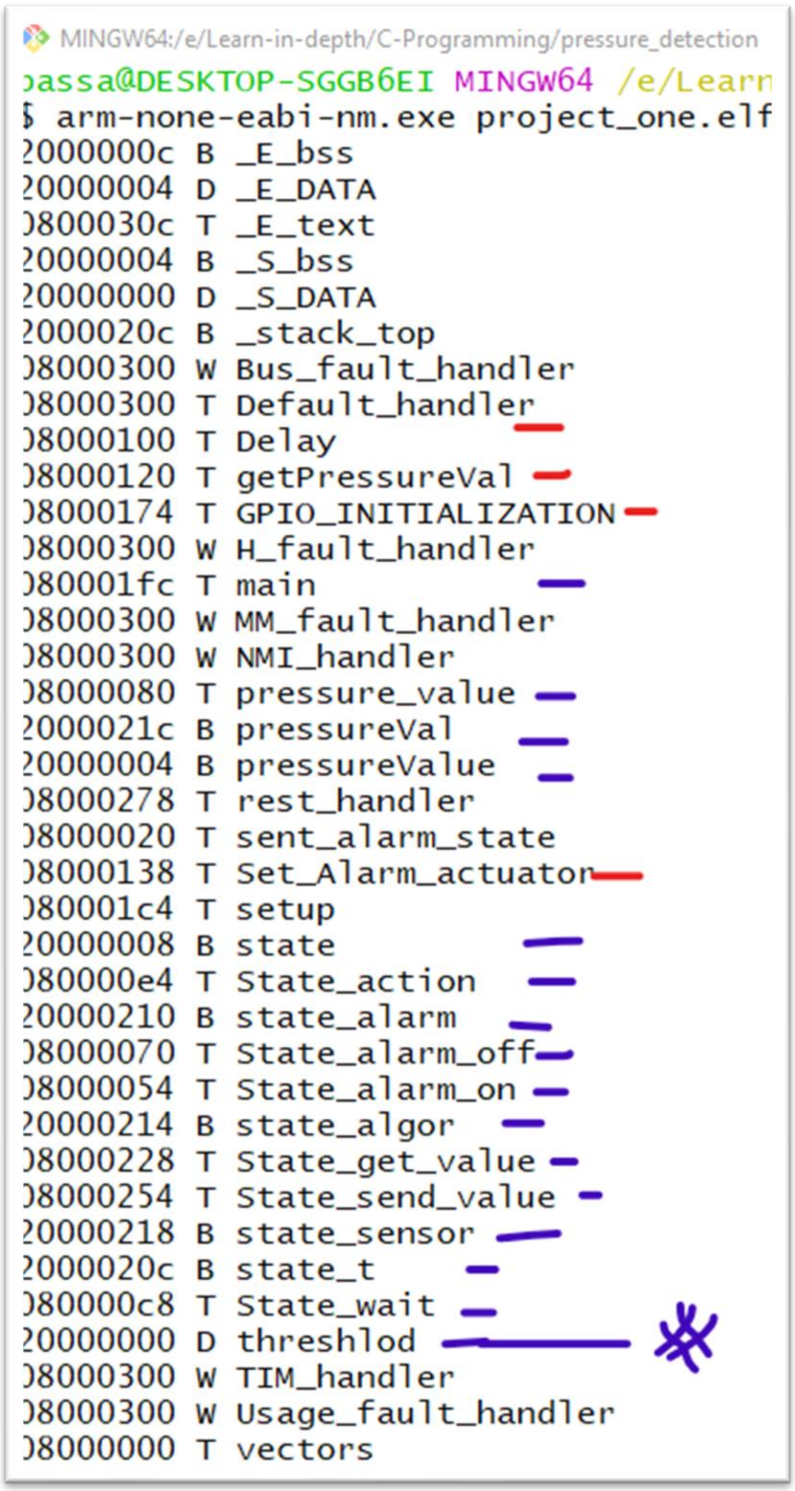
```c
/*
Auther :bassam
Date Feb 5 2022

*/

#include <stdint.h>
#define STACK_POINTER 0x20001000

extern int main(void);
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;
extern uint32_t _stack_top;

void rest_handler(void);
void Default_handler(void);

void rest_handler(void){
    /*copy data section 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;
    for(int i=0; i<Data_size; i++){

        *((uint8_t *)p_dst++) = *((uint8_t *)p_src++);
    }
    /* init .bss with zero*/
    uint32_t bss_size = (uint8_t *)&_E_bss - (uint8_t *)&_S_bss;
    p_dst = (uint8_t *)&_S_bss;
    for(int i=0; i<bss_size; i++){
    *((uint8_t *)p_dst++) = *(uint8_t *)0 ;

    }

 main();
```

count…



```
     main();
}
void Default_handler(void){
     rest_handler();

}
void NMI_handler(void)__attribute__((weak, alias("Default_handler")));
void H_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")));
void TIM_handler(void)__attribute__((weak, alias("Default_handler")));


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

     (uint32_t) &_stack_top,
     (uint32_t) &rest_handler,
     (uint32_t) &NMI_handler,
     (uint32_t) &H_fault_handler,
     (uint32_t) &MM_fault_handler,
     (uint32_t) &Bus_fault_handler,
     (uint32_t) &Usage_fault_handler,
     (uint32_t) &TIM_handler

};
```

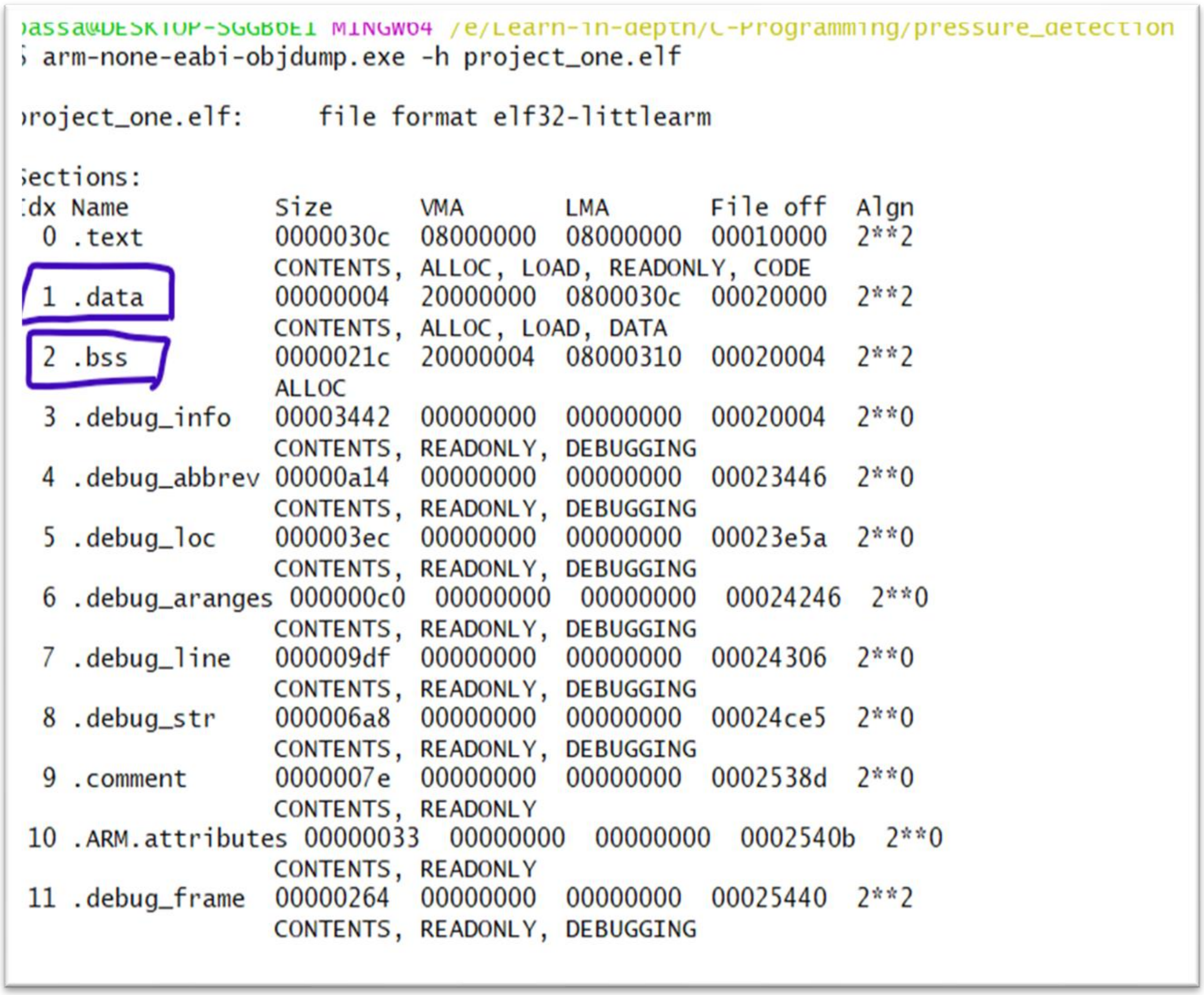
- **software analysis:**
  1- symbols



```
MINGW64:/e/Learn-in-depth/C-Programming/pressure_detection
passa@DESKTOP-SGGB6EI MINGW64 /e/Learn
$ arm-none-eabi-nm.exe project_one.elf
2000000c B _E_bss
20000004 D _E_DATA
0800030c T _E_text
20000004 B _S_bss
20000000 D _S_DATA
2000020c B _stack_top
08000300 W Bus_fault_handler
08000300 T Default_handler
08000100 T Delay
08000120 T getPressureVal
08000174 T GPIO_INITIALIZATION
08000300 W H_fault_handler
080001fc T main
08000300 W MM_fault_handler
08000300 W NMI_handler
08000080 T pressure_value
2000021c B pressureVal
20000004 B pressureValue
08000278 T rest_handler
08000020 T sent_alarm_state
08000138 T Set_Alarm_actuator
080001c4 T setup
20000008 B state
080000e4 T State_action
20000210 B state_alarm
08000070 T State_alarm_off
08000054 T State_alarm_on
20000214 B state_algor
08000228 T State_get_value
08000254 T State_send_value
20000218 B state_sensor
2000020c B state_t
080000c8 T State_wait
20000000 D threshlod
08000300 W TIM_handler
08000300 W Usage_fault_handler
08000000 T vectors
```

- **Threshold at begin of RAM**

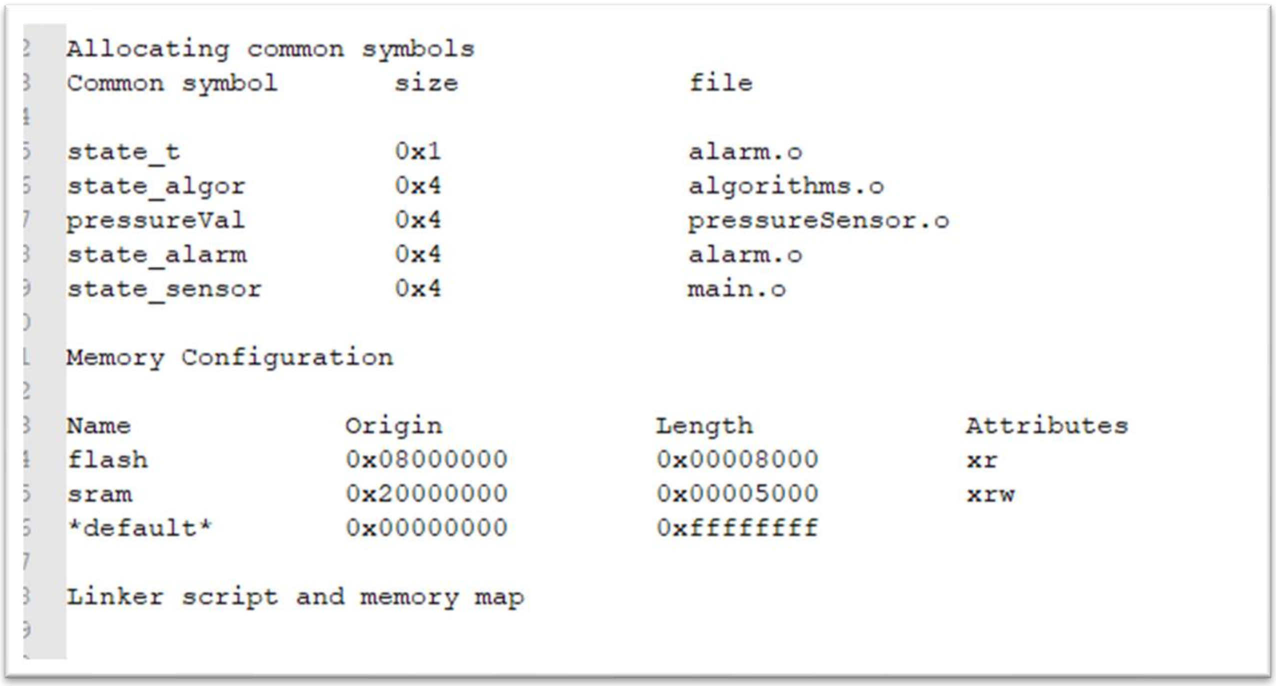
2- Sections:



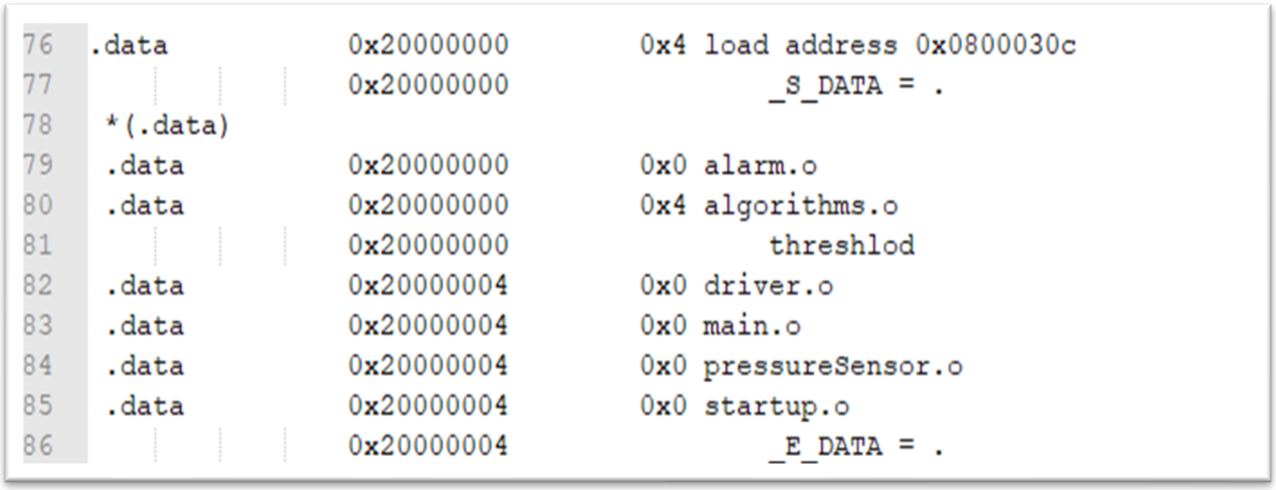
```
)assa@DESKTOP-SGGB6E1 MINGW64 /e/Learn-in-depth/C-Programming/pressure_detection
; arm-none-eabi-objdump.exe -h project_one.elf

)roject_one.elf:     file format elf32-littlearm

;ections:
:dx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000030c  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  20000000  0800030c  00020000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          0000021c  20000004  08000310  00020004  2**2
                  ALLOC
  3 .debug_info   00003442  00000000  00000000  00020004  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 00000a14  00000000  00000000  00023446  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000003ec  00000000  00000000  00023e5a  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000c0  00000000  00000000  00024246  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   000009df  00000000  00000000  00024306  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    000006a8  00000000  00000000  00024ce5  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007e  00000000  00000000  0002538d  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  0002540b  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000264  00000000  00000000  00025440  2**2
                  CONTENTS, READONLY, DEBUGGING
```

- **Dot data section size is 4b only, that is size of threshold variable that I was initialed it with 20, Remaining variables are allocated in bss section.**
- **bss section its size is 0x21c !!!, because stack size in RAM equal 0x200 and we have 7 variable not initialized so, $(7*4)_{\text{decimal}} = (1c)_{\text{hex}}$ Total = 0x200 + 0x1c = 0x21c b.**

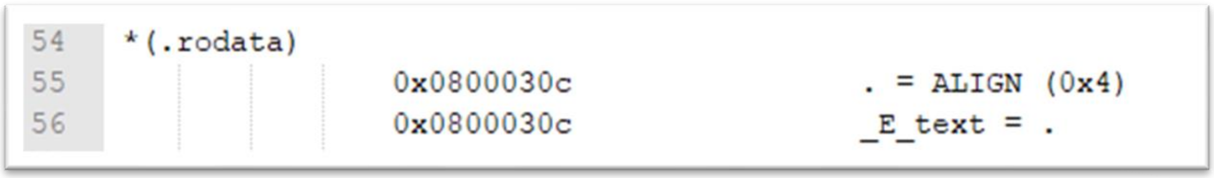
3- Mab file:



```
Allocating common symbols
Common symbol       size              file

state_t             0x1               alarm.o
state_algor         0x4               algorithms.o
pressureVal         0x4               pressureSensor.o
state_alarm         0x4               alarm.o
state_sensor        0x4               main.o

Memory Configuration

Name             Origin             Length             Attributes
flash            0x08000000         0x00008000         xr
sram             0x20000000         0x00005000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map
```



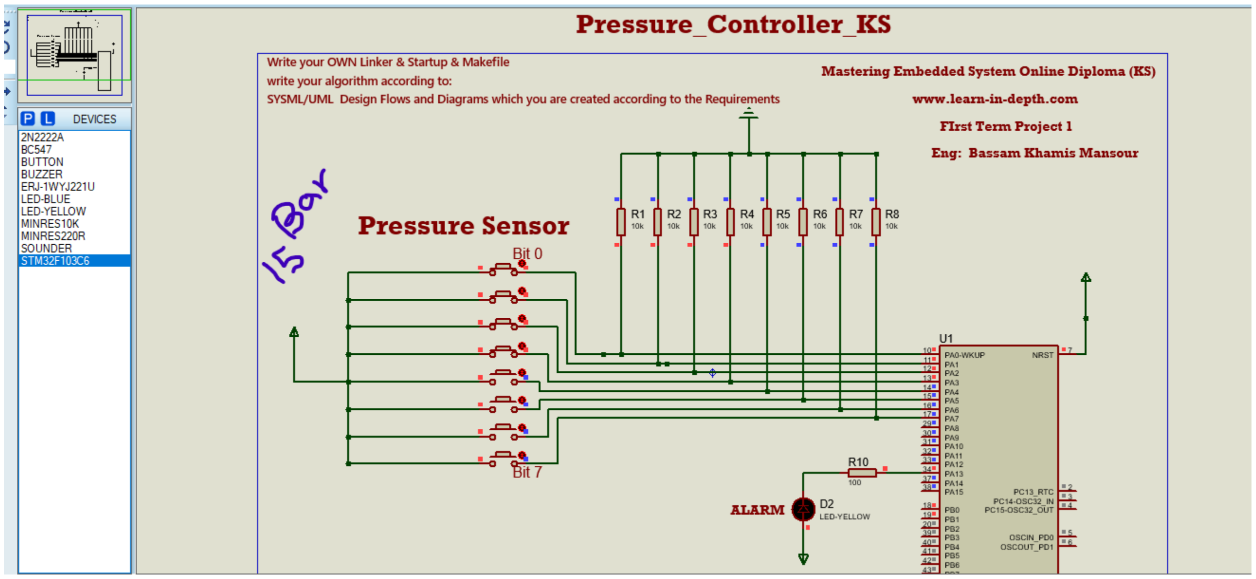
```
.data           0x20000000        0x4 load address 0x0800030c
                0x20000000                _S_DATA = .
 *(.data)
 .data          0x20000000        0x0 alarm.o
 .data          0x20000000        0x4 algorithms.o
                0x20000000                threshlod
 .data          0x20000004        0x0 driver.o
 .data          0x20000004        0x0 main.o
 .data          0x20000004        0x0 pressureSensor.o
 .data          0x20000004        0x0 startup.o
                0x20000004                _E_DATA = .
```



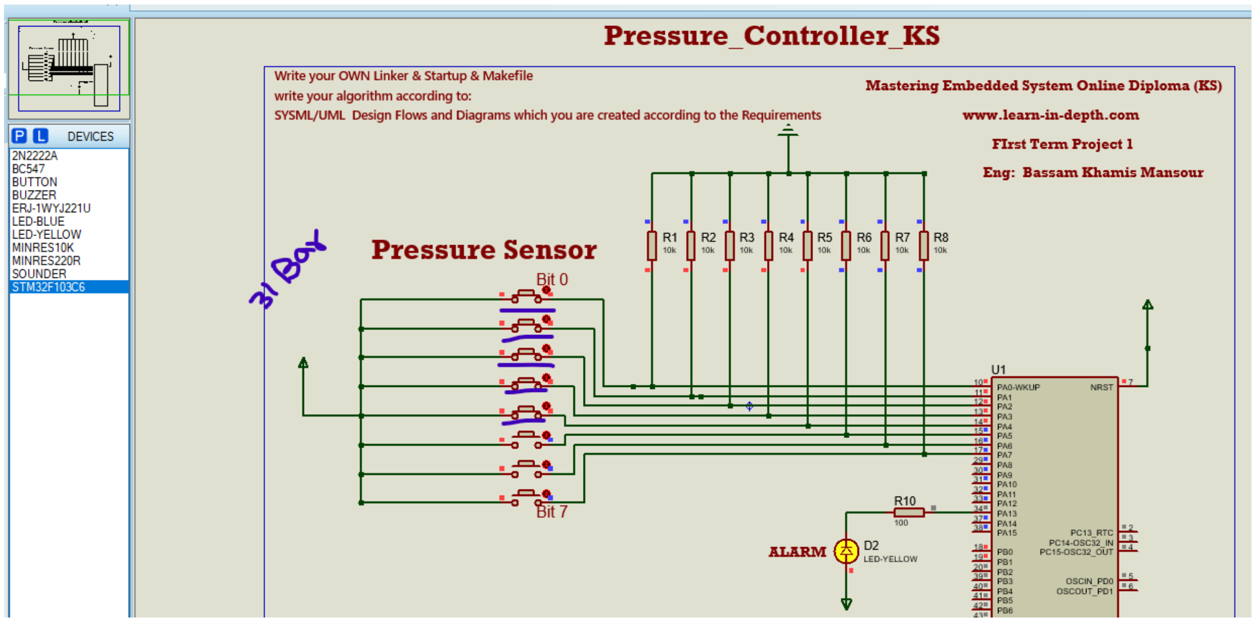
```
 *(.rodata)
                0x0800030c                . = ALIGN (0x4)
                0x0800030c               _E_text = .
```

Finally The output:

Pressure value = 15 Bar less than threshold (20)


Pressure_Controller_KS
Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements
Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Eng: Bassam Khamis Mansour
Pressure Sensor
Bit 0
Bit 7
15 Bar
ALARM
D2
LED-YELLOW
R10
U1
DEVICES
2N2222A
BC547
BUTTON
BUZZER
ERJ-1WYJ221U
LED-BLUE
LED-YELLOW
MINRES10K
MINRES220R
SOUNDER
STM32F103C6

31 Bar greater than 20


Pressure_Controller_KS
Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements
Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Eng: Bassam Khamis Mansour
Pressure Sensor
Bit 0
Bit 7
31 Bar
ALARM
D2
LED-YELLOW
R10
U1
DEVICES
STM32F103C6

Thank you……