Pressure Controller Project

Learn In Depth

Eng . kerolous shenoda

Name	Roaa Aiman Fahmy

Table of contents

Case Study

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. Keep track of the measured values.

Assumptions

- The system setup and shutdown procedures are not modeled.

- The system maintenance is not modeled.

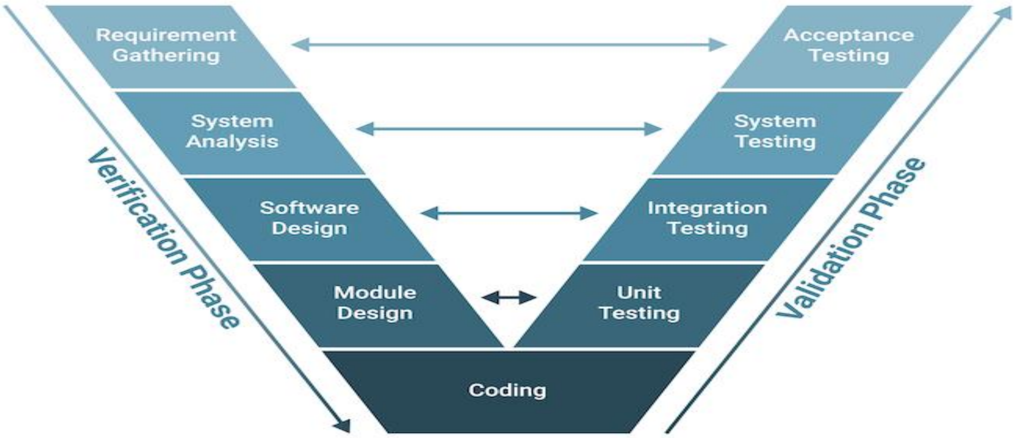
- The pressure sensor never fails.

- The alarm never fails.

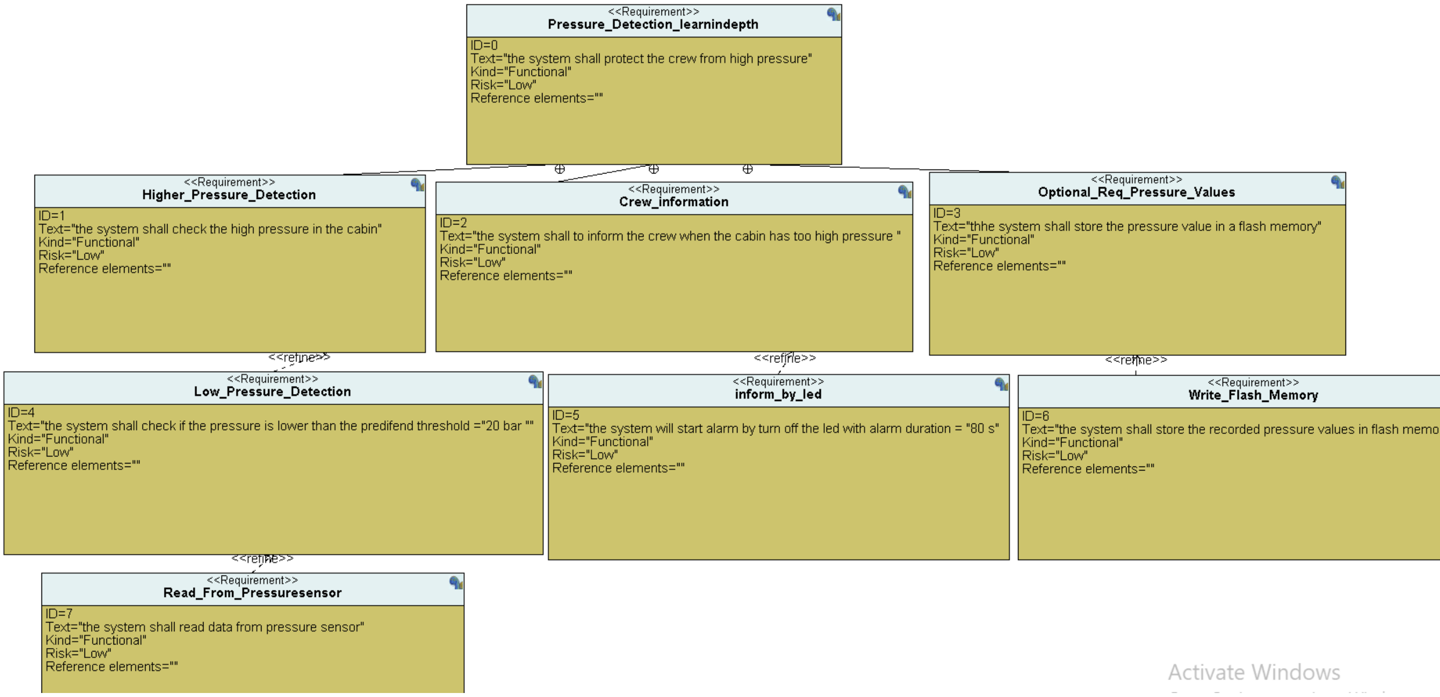
- The system never faces power cut.

Methodology

Since the system has multiple modules that are no easy to integrate, the system will use a testing-based model like v-model. Every phase in this project will be tested and especially the implementation phase. Each software module will be implemented and unit-tested separately then integrated and integration testing will be performed.



Requirement Diagram



<<Requirement>>
Pressure_Detection_learnindepth
ID=0
Text="the system shall protect the crew from high pressure"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Higher_Pressure_Detection
ID=1
Text="the system shall check the high pressure in the cabin"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Crew_information
ID=2
Text="the system shall to inform the crew when the cabin has too high pressure "
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Optional_Req_Pressure_Values
ID=3
Text="thhe system shall store the pressure value in a flash memory"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Low_Pressure_Detection
ID=4
Text="the system shall check if the pressure is lower than the predifend threshold ="20 bar ""
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
inform_by_led
ID=5
Text="the system will start alarm by turn off the led with alarm duration = "80 s""
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Write_Flash_Memory
ID=6
Text="the system shall store the recorded pressure values in flash memo
Kind="Functional"
Risk="Low"
Reference elements=""

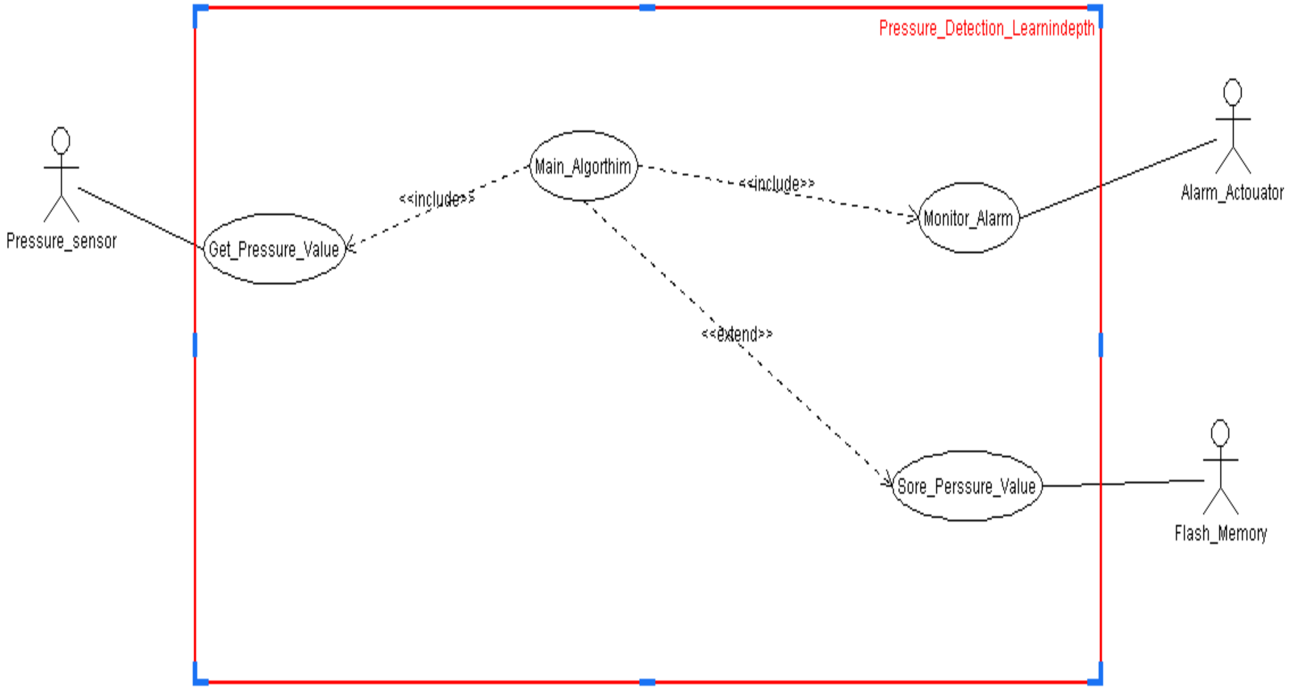
<<Requirement>>
Read_From_Pressuresensor
ID=7
Text="the system shall read data from pressure sensor"
Kind="Functional"
Risk="Low"
Reference elements=""

Space Exploration (HW/SW Partitioning)

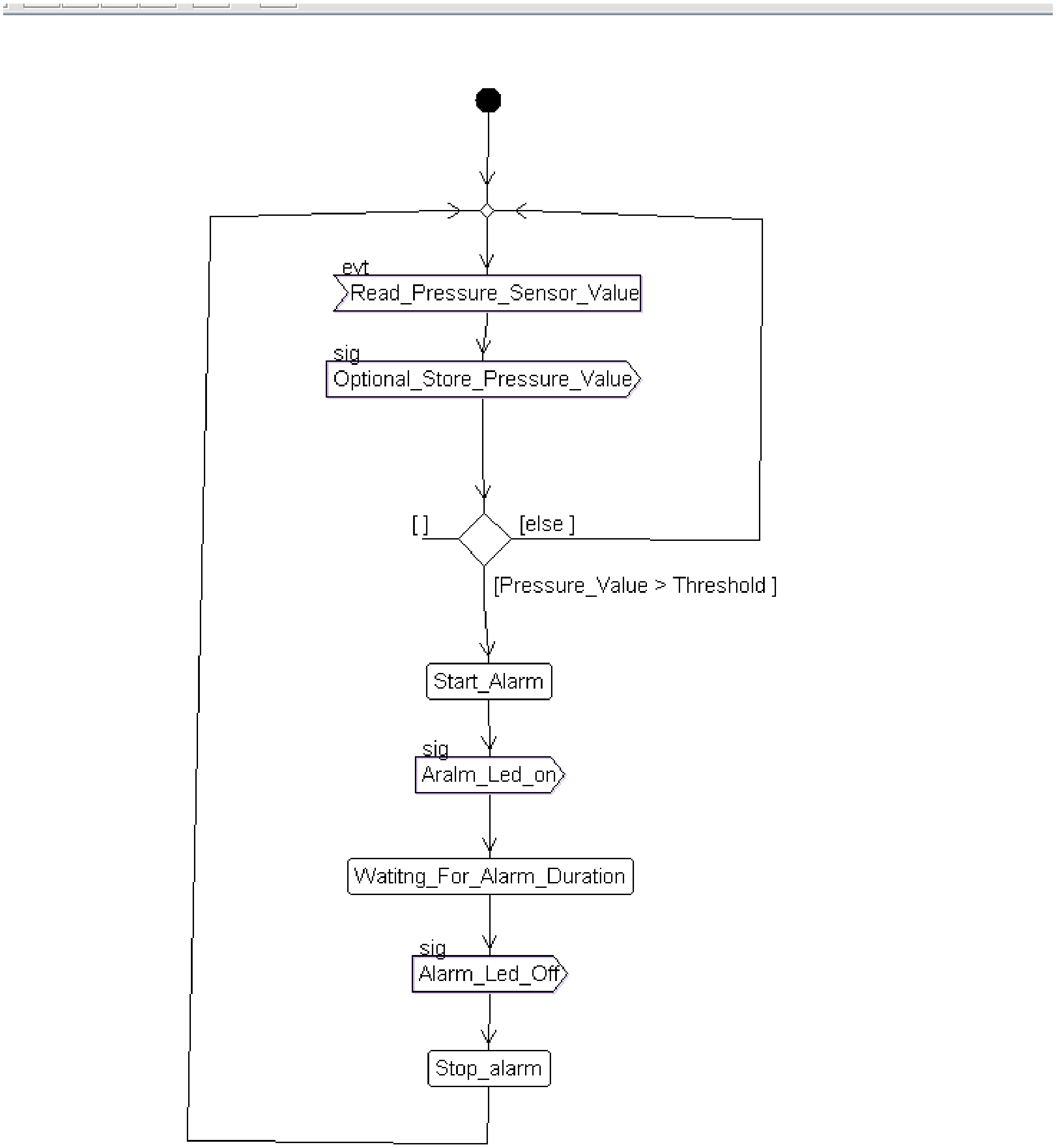
For the hardware, we have STM32103c6 microcontroller with a cortex-m3 processor based on ARM

System Analysis

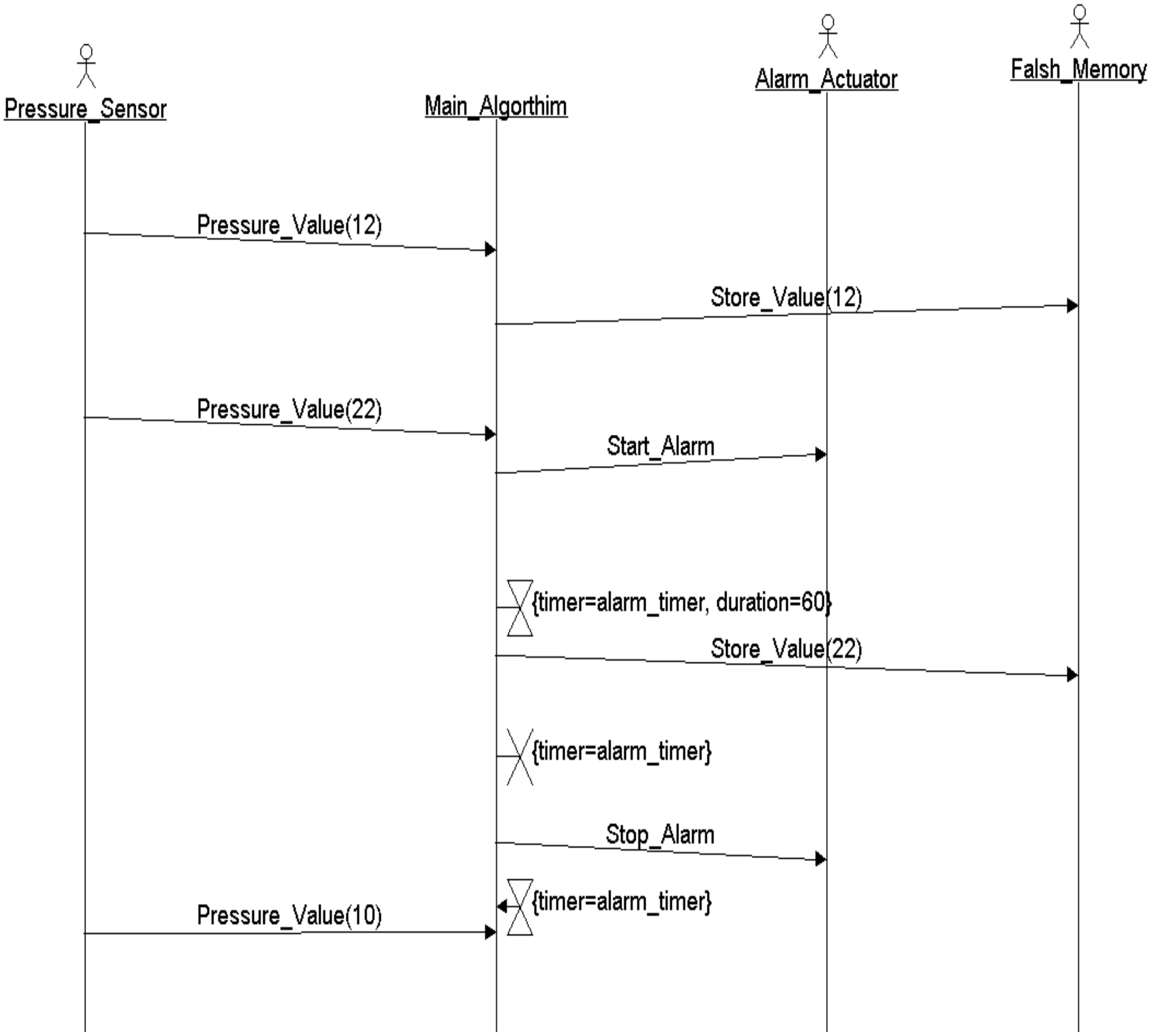
- ## Use Case Diagram



• Activity Diagram

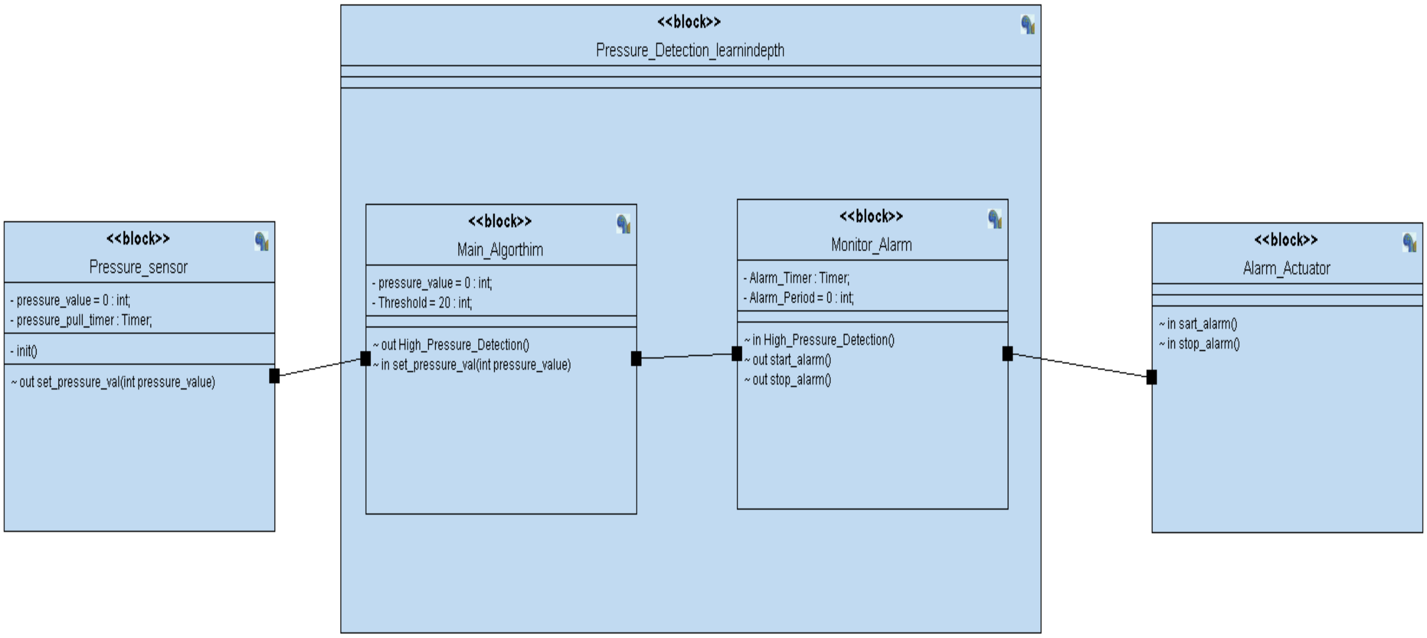

evt
Read_Pressure_Sensor_Value
sig
Optional_Store_Pressure_Value
[ ]
[else]
[Pressure_Value > Threshold]
Start_Alarm
sig
Aralm_Led_on
Watitng_For_Alarm_Duration
sig
Alarm_Led_Off
Stop_alarm

- ## Sequence Diagram

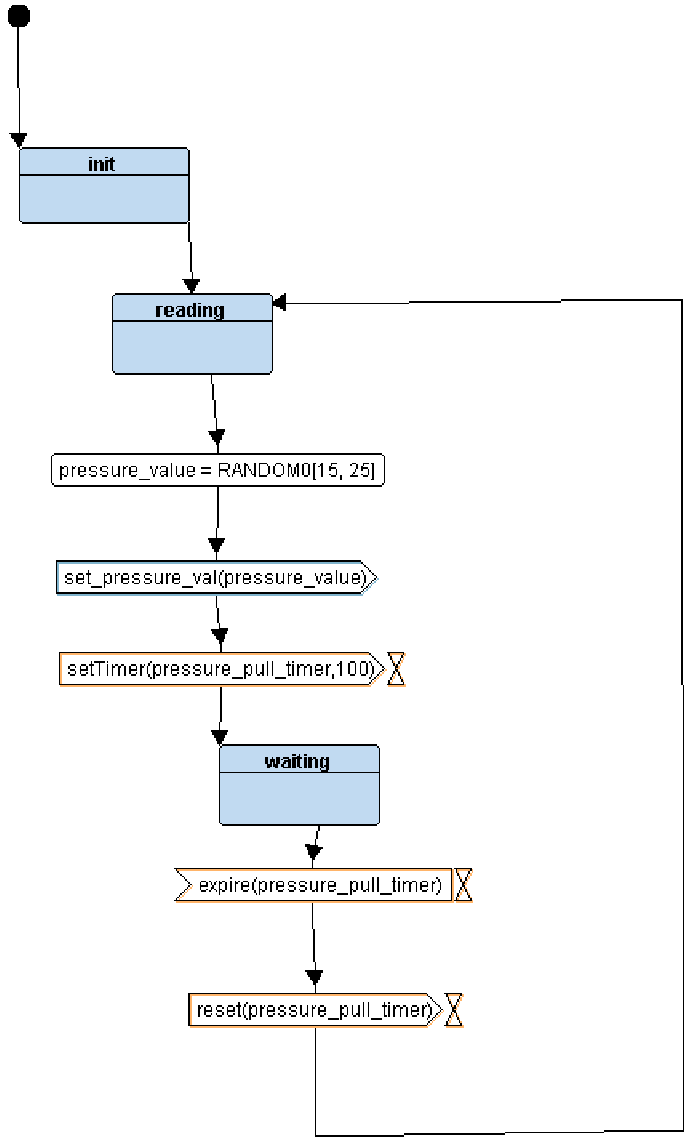


System Design

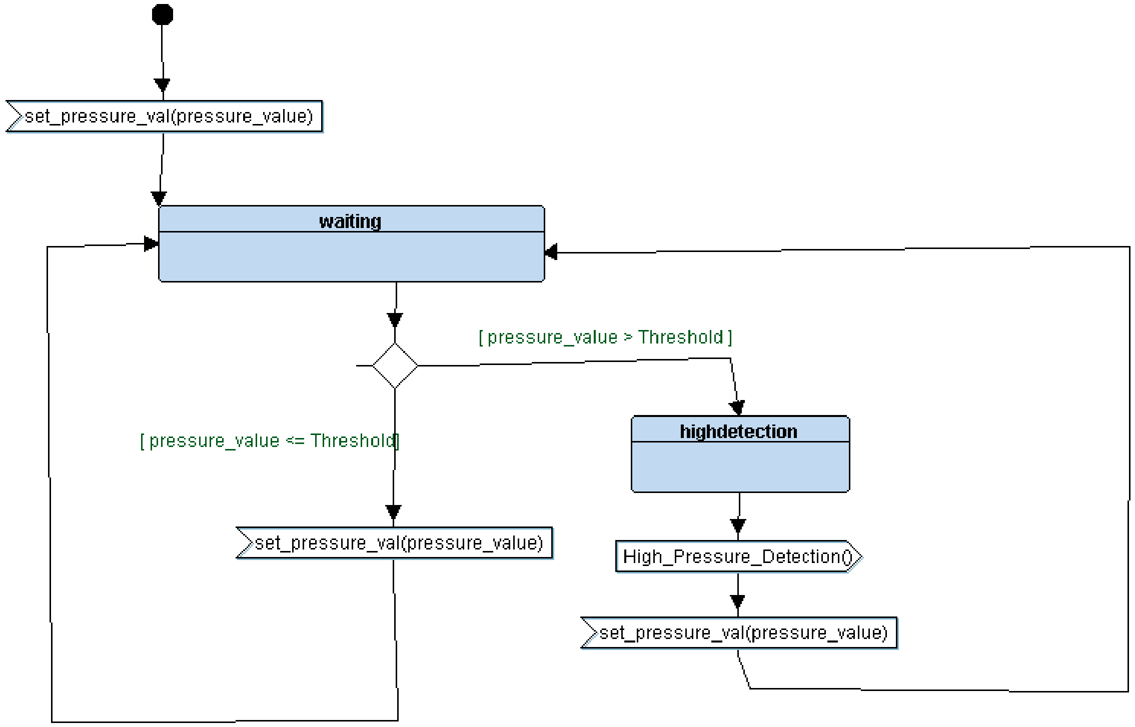
• Block Diagram


<<block>>
Pressure_Detection_learnindepth
<<block>>
Pressure_sensor
- pressure_value = 0 : int;
- pressure_pull_timer : Timer;
- init()
~ out set_pressure_val(int pressure_value)
<<block>>
Main_Algorthim
- pressure_value = 0 : int;
- Threshold = 20 : int;
~ out High_Pressure_Detection()
~ in set_pressure_val(int pressure_value)
<<block>>
Monitor_Alarm
- Alarm_Timer : Timer;
- Alarm_Period = 0 : int;
~ in High_Pressure_Detection()
~ out start_alarm()
~ out stop_alarm()
<<block>>
Alarm_Actuator
~ in sart_alarm()
~ in stop_alarm()

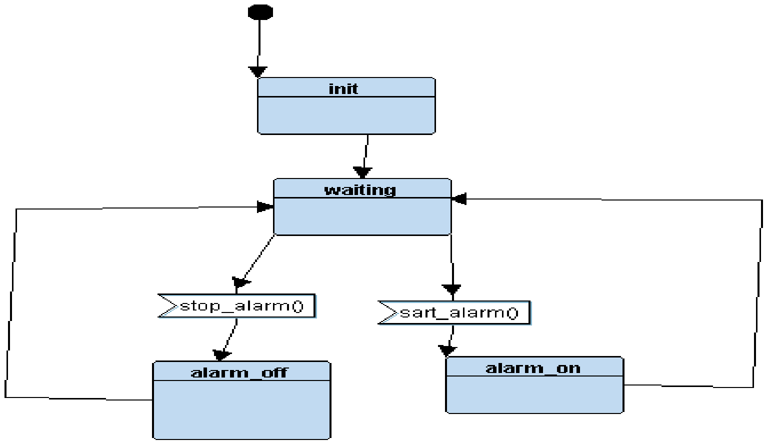
- **Pressure Sensor State Diagram**


init
reading
pressure_value = RANDOM0[15, 25]
set_pressure_val(pressure_value)
setTimer(pressure_pull_timer,100)
waiting
expire(pressure_pull_timer)
reset(pressure_pull_timer)

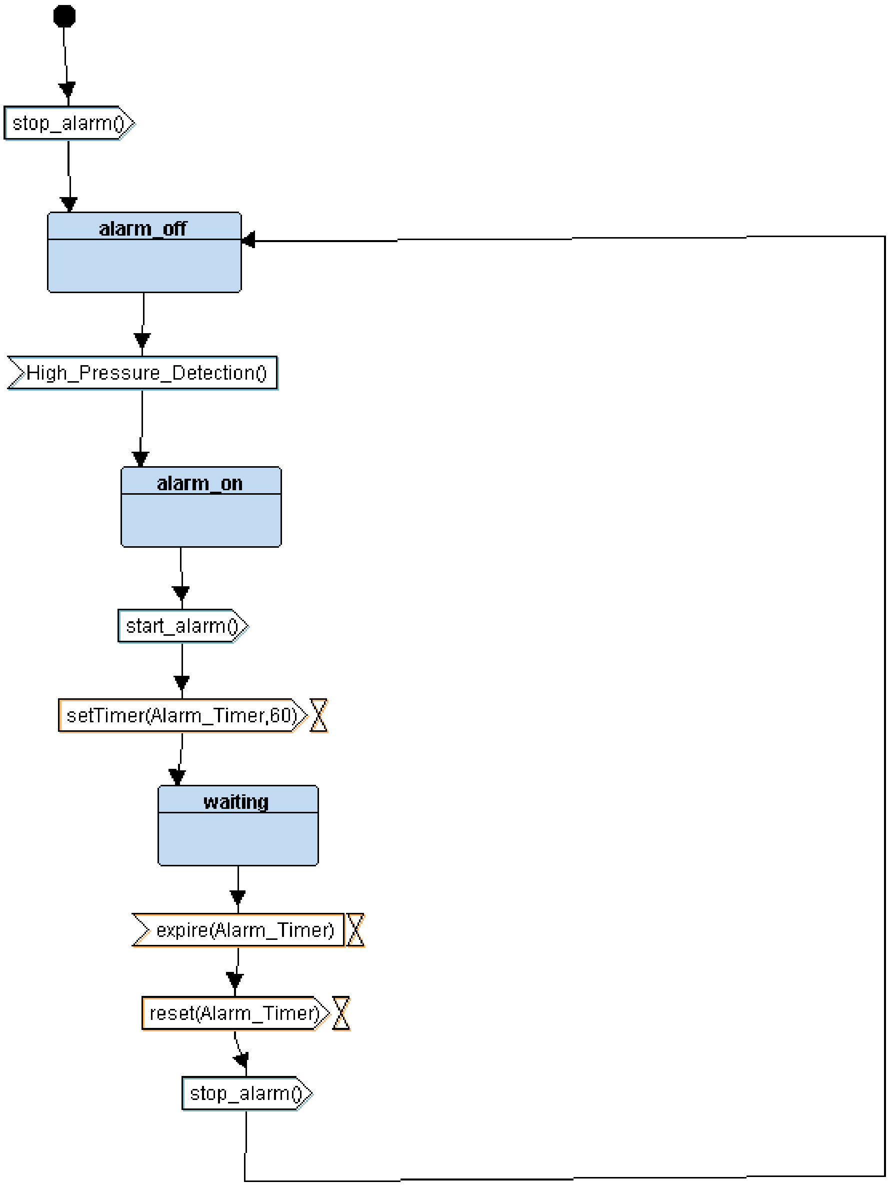
- # Main Program State Diagram


set_pressure_val(pressure_value)
waiting
[pressure_value > Threshold]
[pressure_value <= Threshold]
highdetection
set_pressure_val(pressure_value)
High_Pressure_Detection()
set_pressure_val(pressure_value)

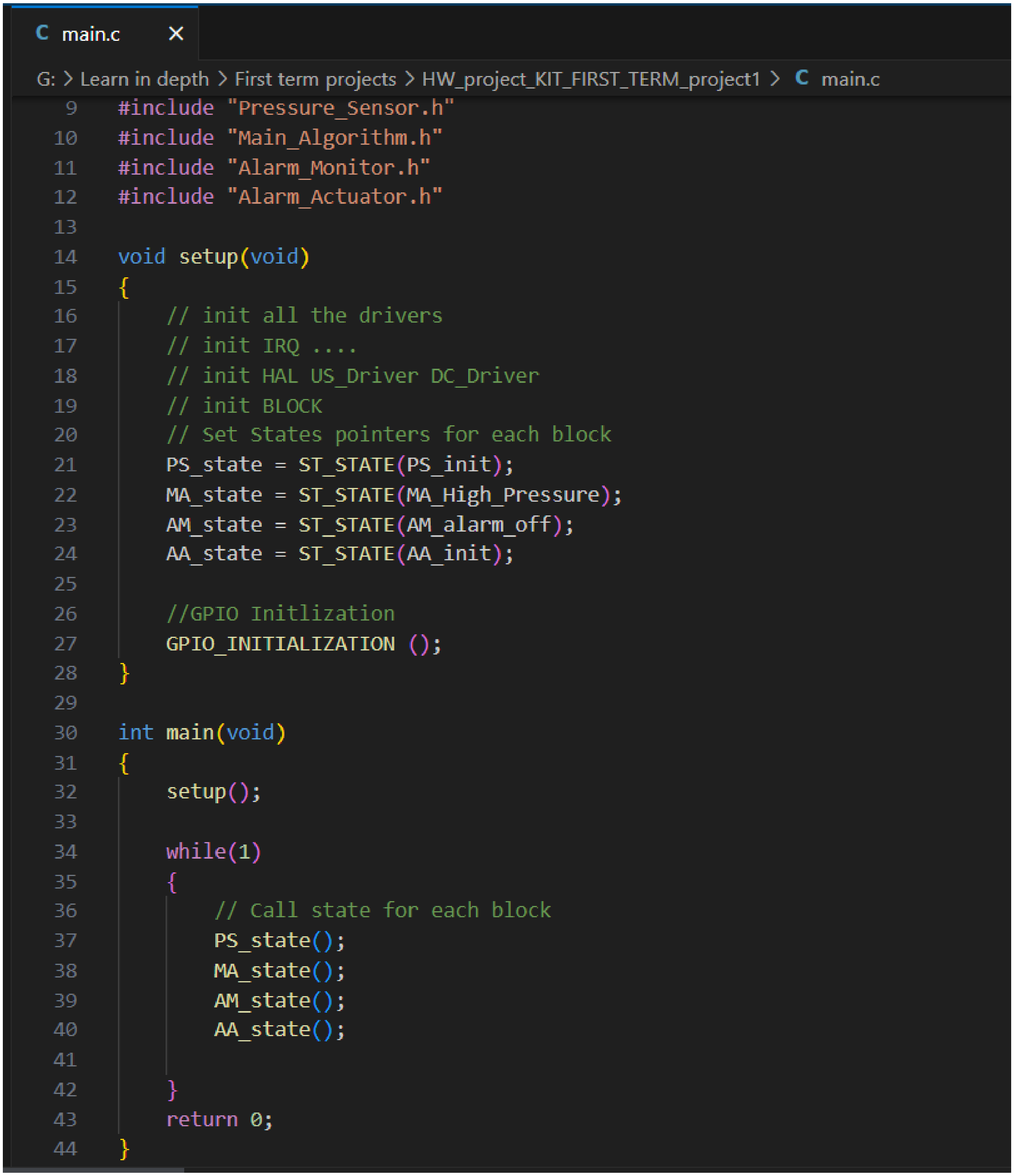
- # Alarm Actuator State Diagram



- Alarm Monitor State Diagram

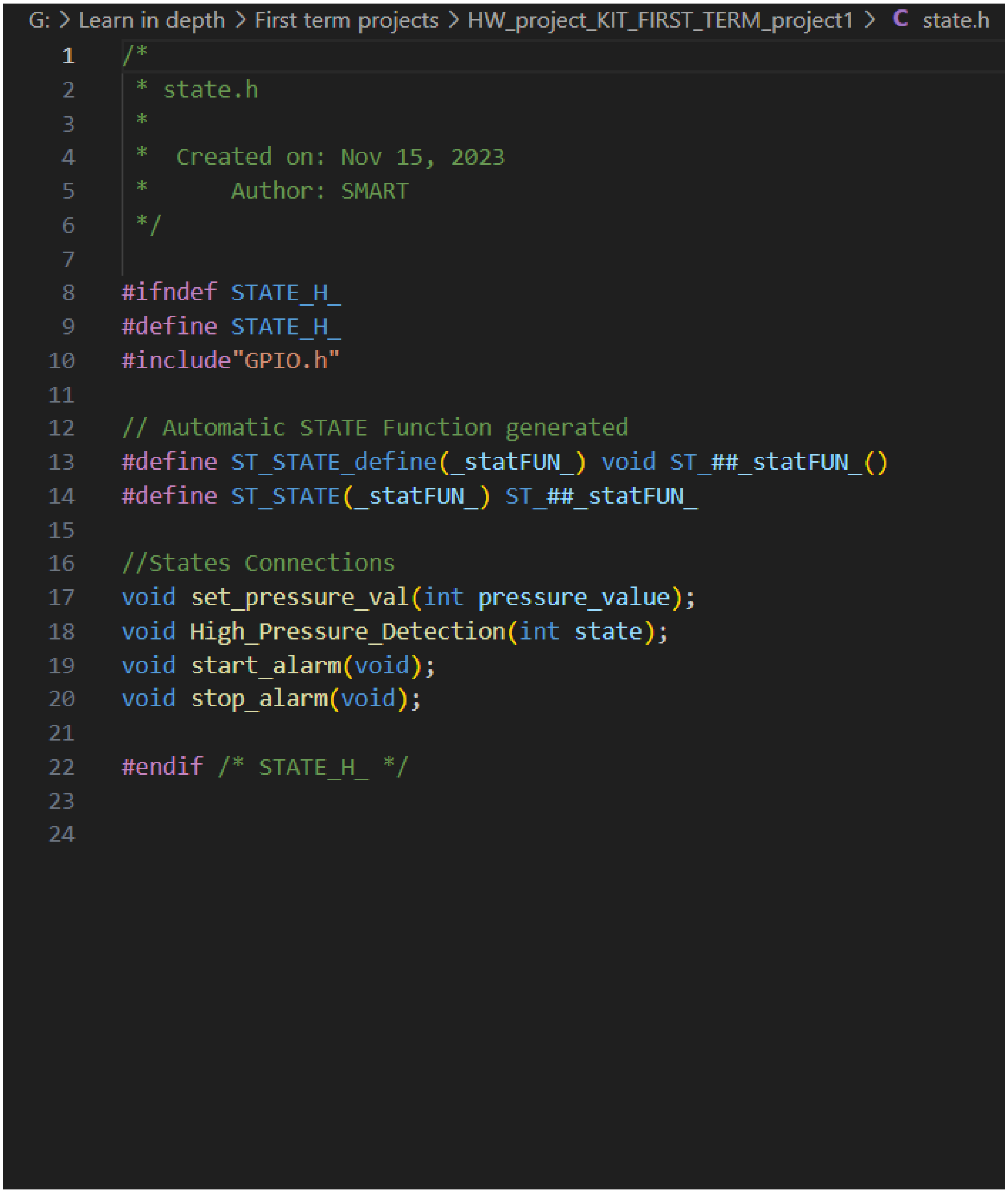


Main.c file


main.c
G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > main.c

```c
#include "Pressure_Sensor.h"
#include "Main_Algorithm.h"
#include "Alarm_Monitor.h"
#include "Alarm_Actuator.h"

void setup(void)
{
    // init all the drivers
    // init IRQ ....
    // init HAL US_Driver DC_Driver
    // init BLOCK
    // Set States pointers for each block
    PS_state = ST_STATE(PS_init);
    MA_state = ST_STATE(MA_High_Pressure);
    AM_state = ST_STATE(AM_alarm_off);
    AA_state = ST_STATE(AA_init);

    //GPIO Initlization
    GPIO_INITIALIZATION ();
}

int main(void)
{
    setup();

    while(1)
    {
        // Call state for each block
        PS_state();
        MA_state();
        AM_state();
        AA_state();

    }
    return 0;
}
```

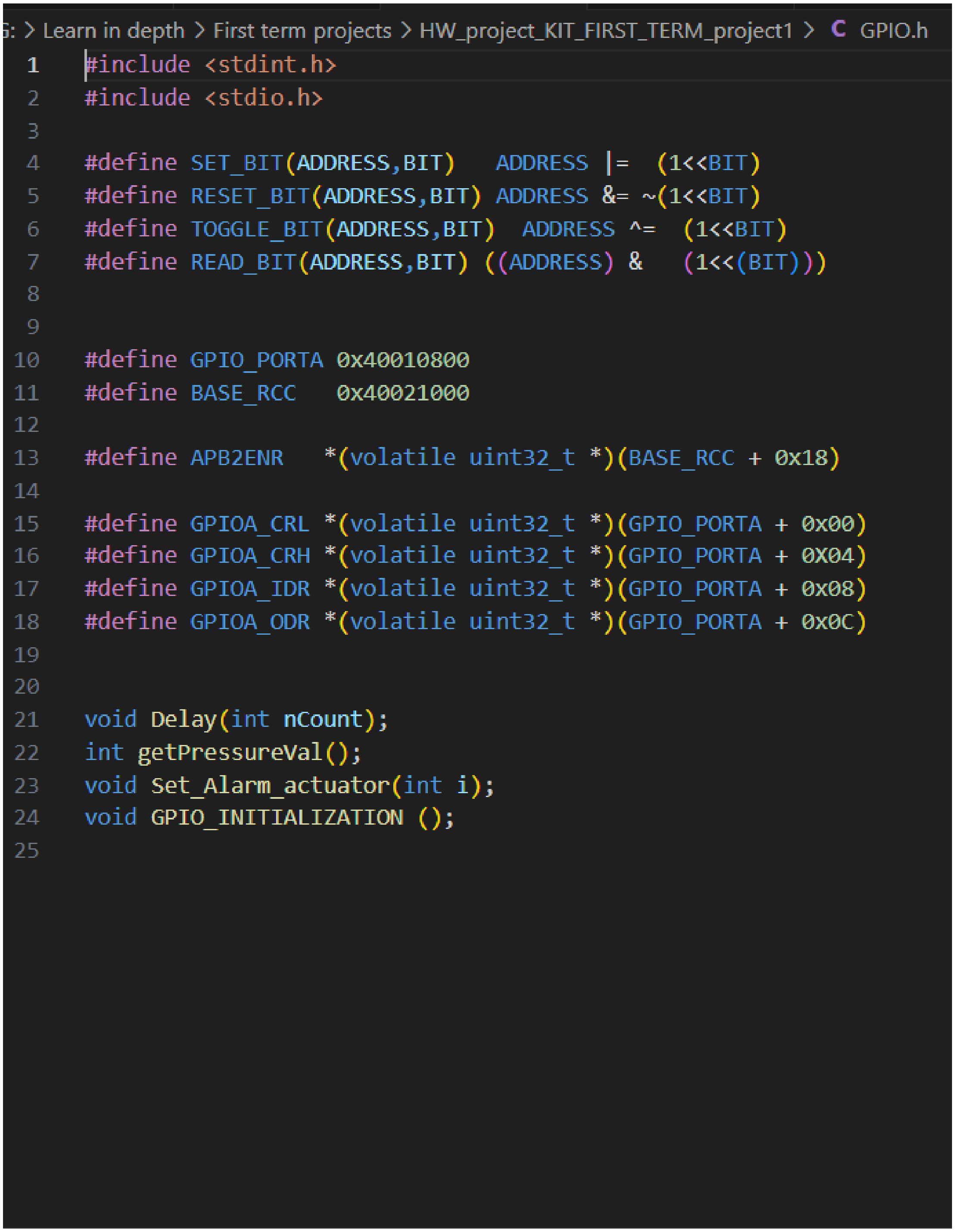
State.h file


G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > C state.h

```c
/*
 * state.h
 *
 *  Created on: Nov 15, 2023
 *      Author: SMART
 */

#ifndef STATE_H_
#define STATE_H_
#include"GPIO.h"

// Automatic STATE Function generated
#define ST_STATE_define(_statFUN_) void ST_##_statFUN_()
#define ST_STATE(_statFUN_) ST_##_statFUN_

//States Connections
void set_pressure_val(int pressure_value);
void High_Pressure_Detection(int state);
void start_alarm(void);
void stop_alarm(void);

#endif /* STATE_H_ */
```

GPIO

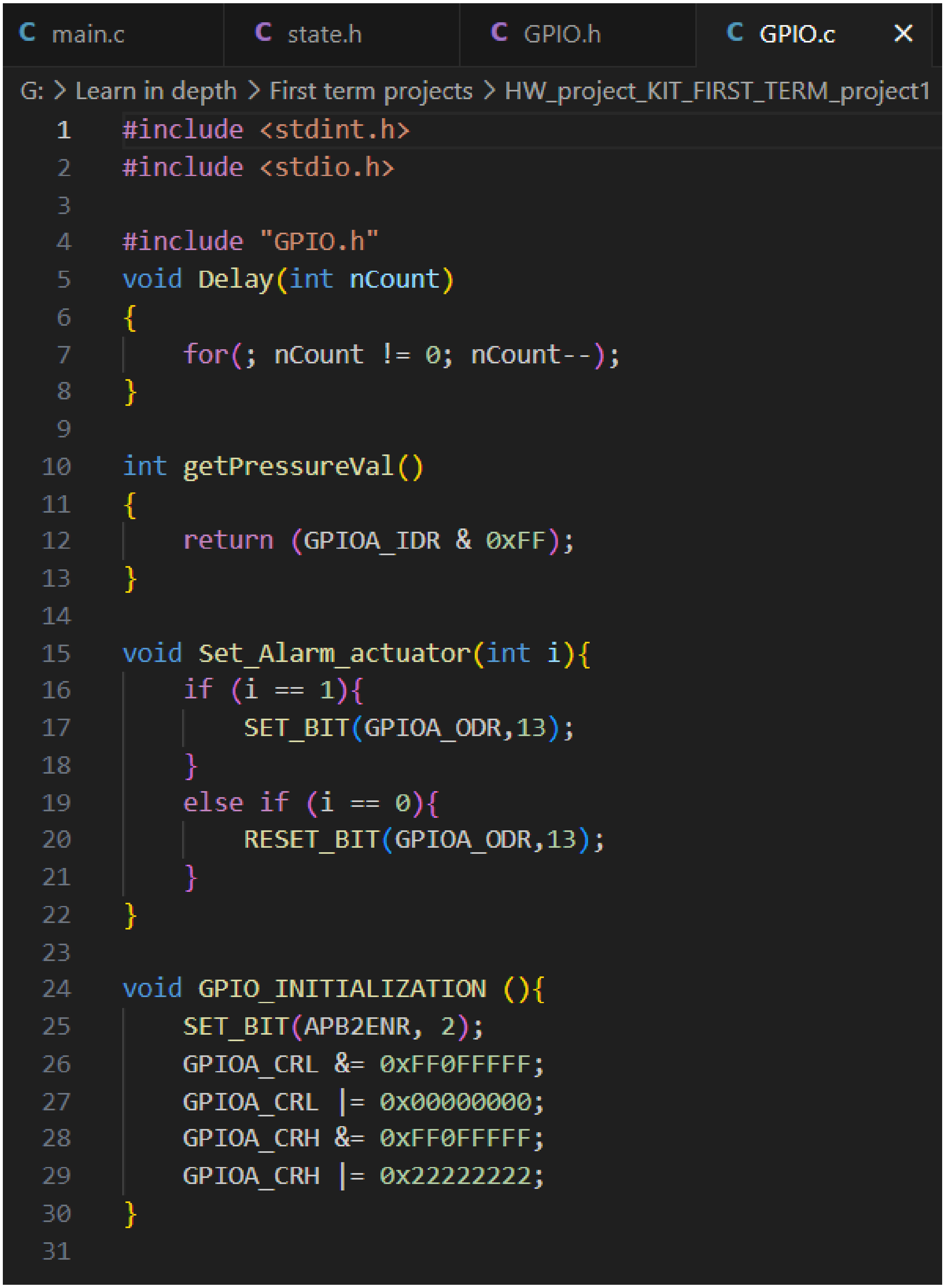
- ## GPIO.h



: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > C GPIO.h

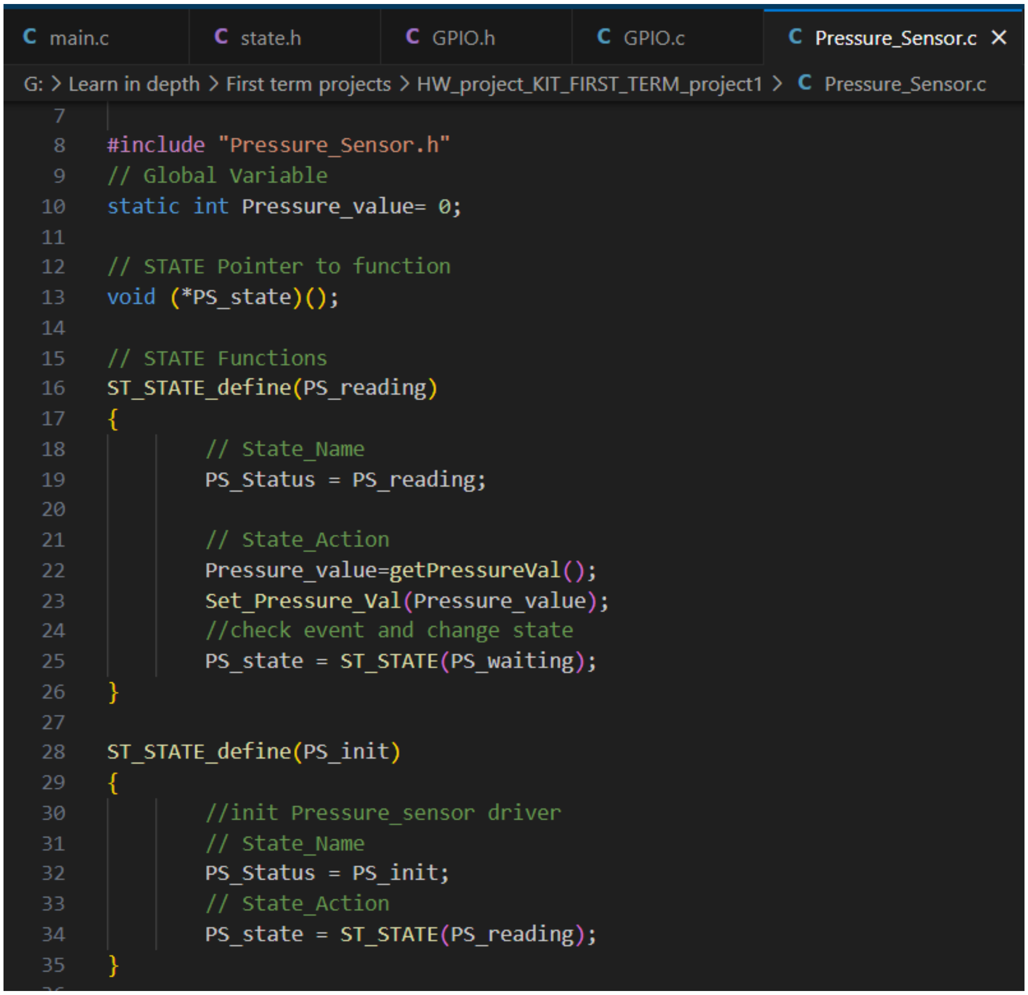
```c
#include <stdint.h>
#include <stdio.h>

#define SET_BIT(ADDRESS,BIT)   ADDRESS |=  (1<<BIT)
#define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
#define TOGGLE_BIT(ADDRESS,BIT)  ADDRESS ^=  (1<<BIT)
#define READ_BIT(ADDRESS,BIT) ((ADDRESS) &   (1<<(BIT)))


#define GPIO_PORTA 0x40010800
#define BASE_RCC   0x40021000

#define APB2ENR   *(volatile uint32_t *)(BASE_RCC + 0x18)

#define GPIOA_CRL *(volatile uint32_t *)(GPIO_PORTA + 0x00)
#define GPIOA_CRH *(volatile uint32_t *)(GPIO_PORTA + 0X04)
#define GPIOA_IDR *(volatile uint32_t *)(GPIO_PORTA + 0x08)
#define GPIOA_ODR *(volatile uint32_t *)(GPIO_PORTA + 0x0C)


void Delay(int nCount);
int getPressureVal();
void Set_Alarm_actuator(int i);
void GPIO_INITIALIZATION ();

```

- GPIO.C

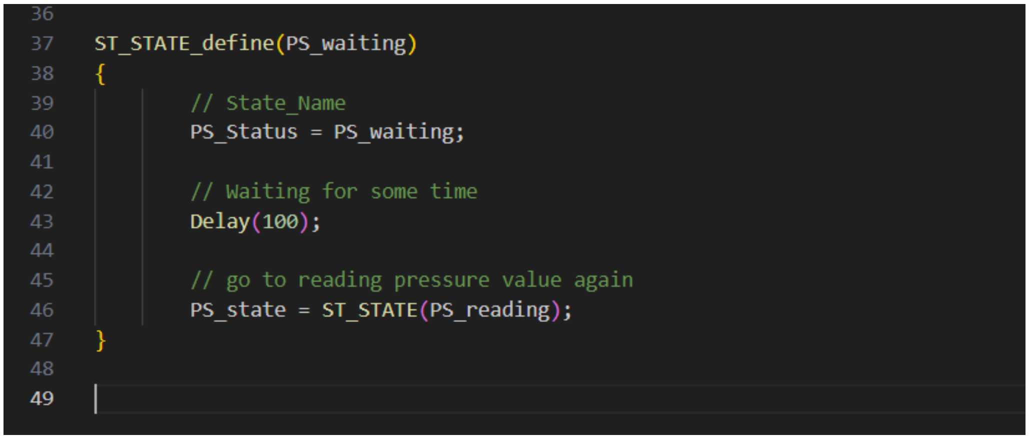

C main.c
C state.h
C GPIO.h
C GPIO.c ✕
G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1

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

#include "GPIO.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;
}

```

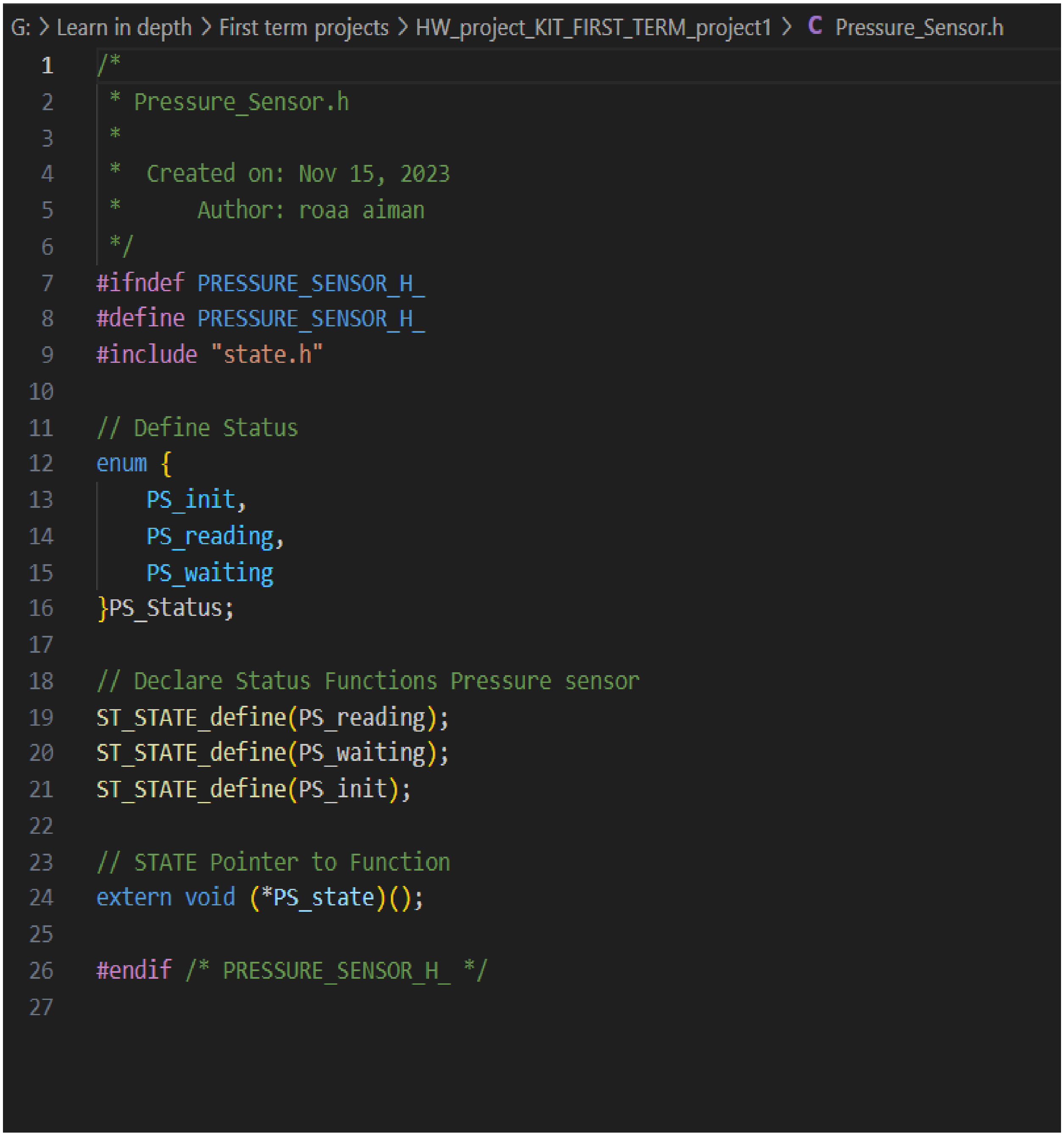
Pressure Sensor


main.c
state.h
GPIO.h
GPIO.c
Pressure_Sensor.c
G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > Pressure_Sensor.c

```c

#include "Pressure_Sensor.h"
// Global Variable
static int Pressure_value= 0;

// STATE Pointer to function
void (*PS_state)();

// STATE Functions
ST_STATE_define(PS_reading)
{
        // State_Name
        PS_Status = PS_reading;

        // State_Action
        Pressure_value=getPressureVal();
        Set_Pressure_Val(Pressure_value);
        //check event and change state
        PS_state = ST_STATE(PS_waiting);
}

ST_STATE_define(PS_init)
{
        //init Pressure_sensor driver
        // State_Name
        PS_Status = PS_init;
        // State_Action
        PS_state = ST_STATE(PS_reading);
}
```



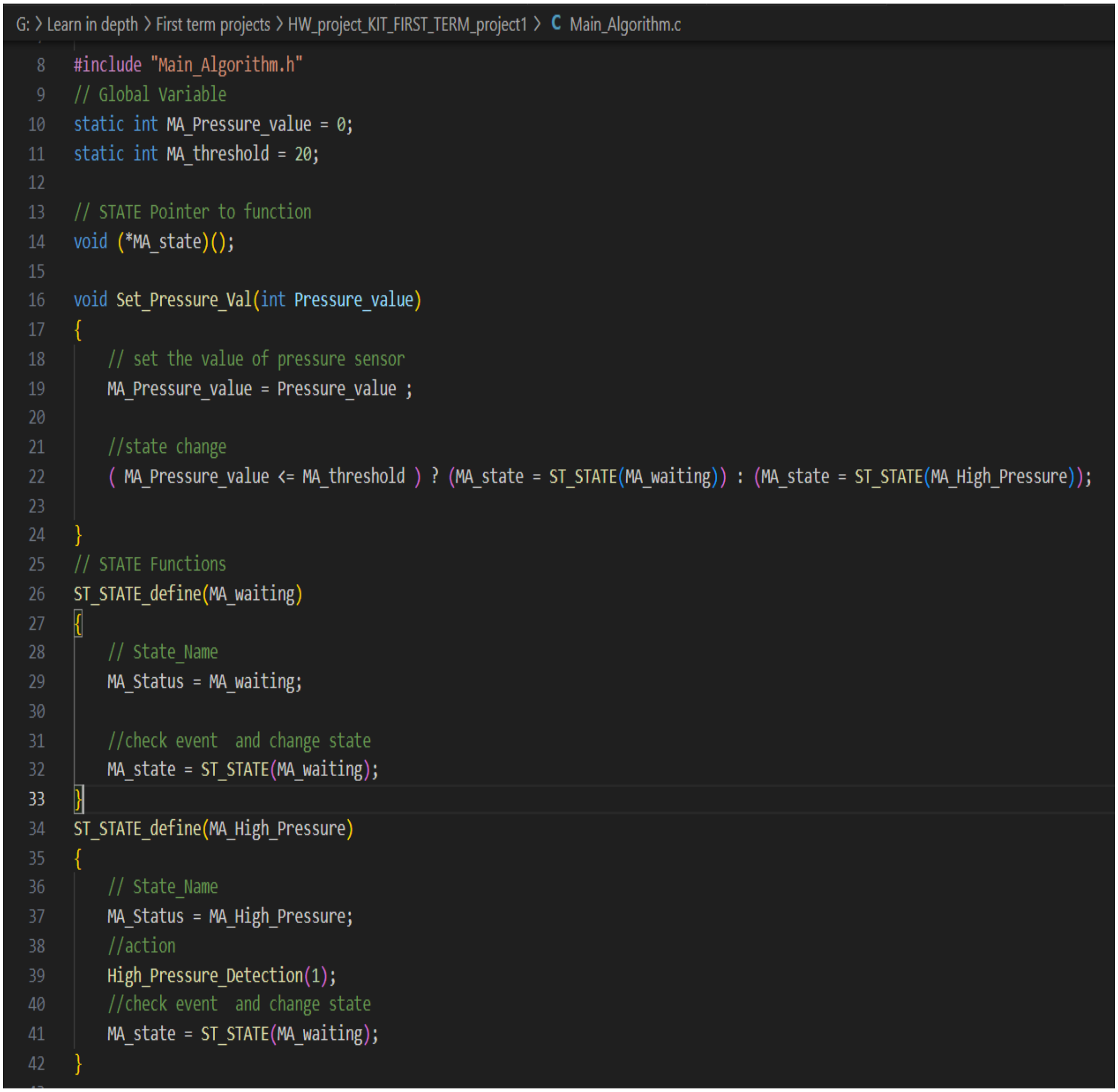
```c

ST_STATE_define(PS_waiting)
{
        // State_Name
        PS_Status = PS_waiting;

        // Waiting for some time
        Delay(100);

        // go to reading pressure value again
        PS_state = ST_STATE(PS_reading);
}


```



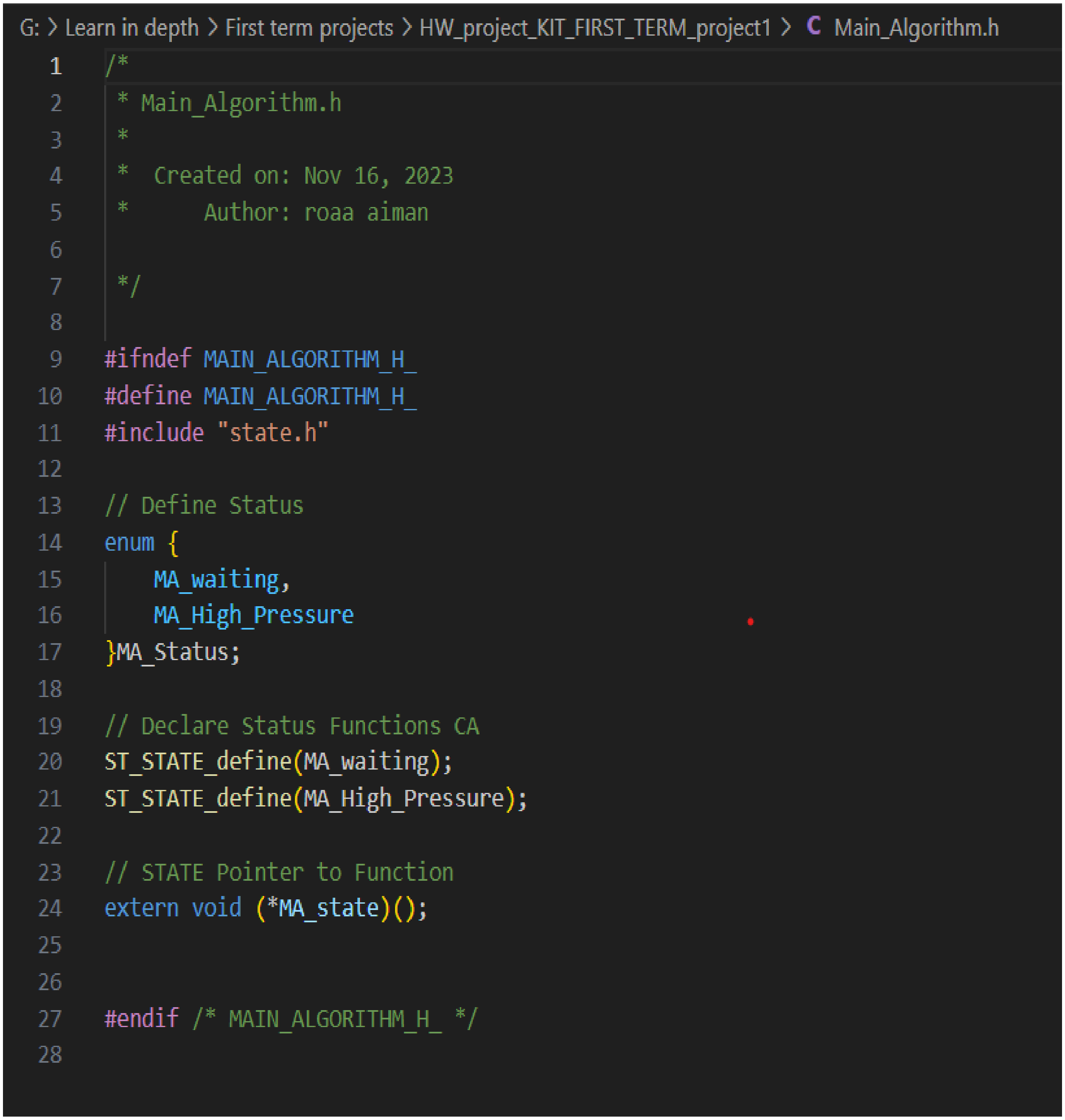
G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > Pressure_Sensor.h

```c
/*
 * Pressure_Sensor.h
 *
 *  Created on: Nov 15, 2023
 *      Author: roaa aiman
 */
#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_
#include "state.h"

// Define Status
enum {
    PS_init,
    PS_reading,
    PS_waiting
}PS_Status;

// Declare Status Functions Pressure sensor
ST_STATE_define(PS_reading);
ST_STATE_define(PS_waiting);
ST_STATE_define(PS_init);

// STATE Pointer to Function
extern void (*PS_state)();

#endif /* PRESSURE_SENSOR_H_ */

```

Main Algorthim

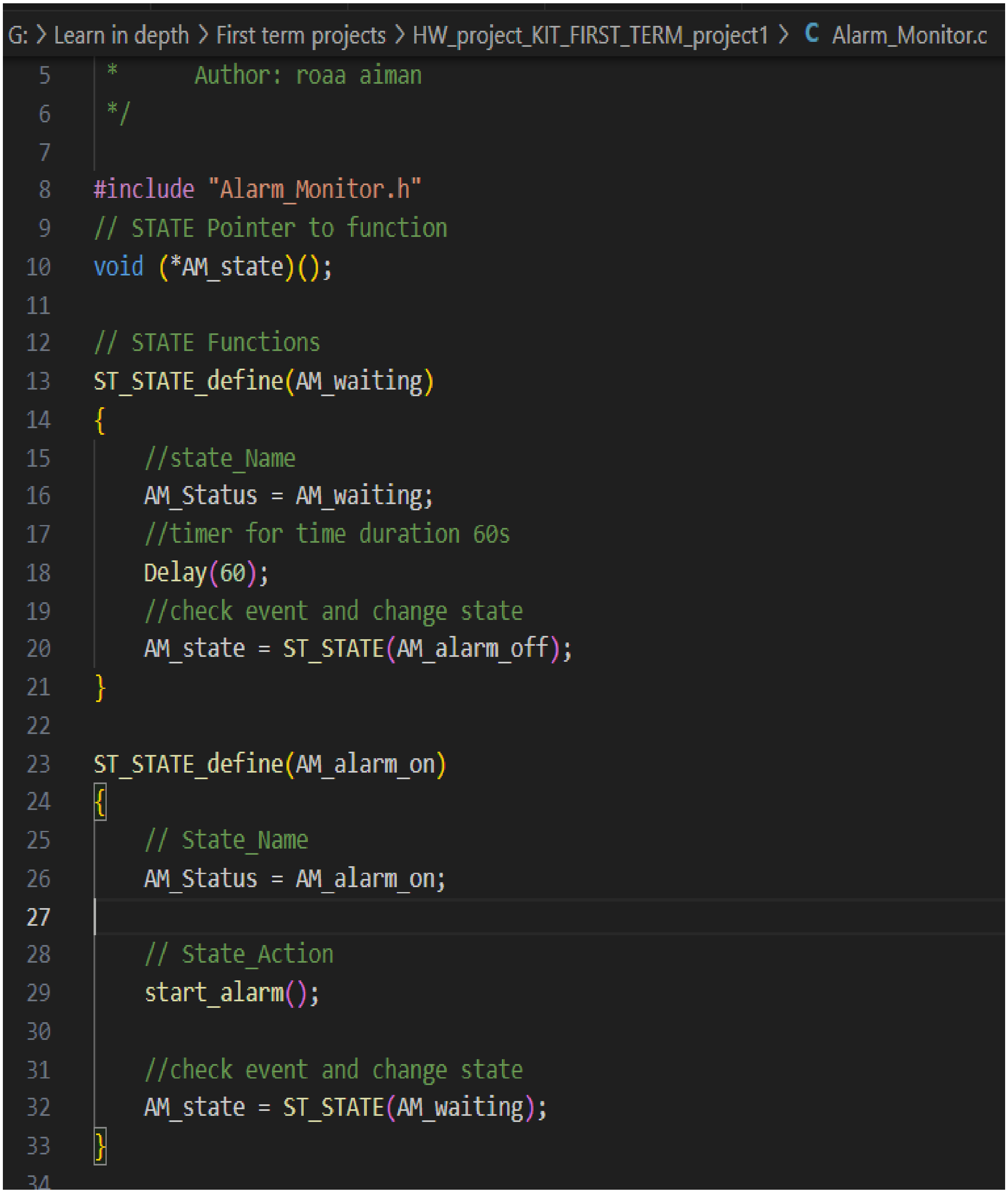

G: › Learn in depth › First term projects › HW_project_KIT_FIRST_TERM_project1 › C Main_Algorithm.c

```c
#include "Main_Algorithm.h"
// Global Variable
static int MA_Pressure_value = 0;
static int MA_threshold = 20;

// STATE Pointer to function
void (*MA_state)();

void Set_Pressure_Val(int Pressure_value)
{
    // set the value of pressure sensor
    MA_Pressure_value = Pressure_value ;

    //state change
    ( MA_Pressure_value <= MA_threshold ) ? (MA_state = ST_STATE(MA_waiting)) : (MA_state = ST_STATE(MA_High_Pressure));

}
// STATE Functions
ST_STATE_define(MA_waiting)
{
    // State_Name
    MA_Status = MA_waiting;

    //check event  and change state
    MA_state = ST_STATE(MA_waiting);
}
ST_STATE_define(MA_High_Pressure)
{
    // State_Name
    MA_Status = MA_High_Pressure;
    //action
    High_Pressure_Detection(1);
    //check event  and change state
    MA_state = ST_STATE(MA_waiting);
}
```


G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > Main_Algorithm.h

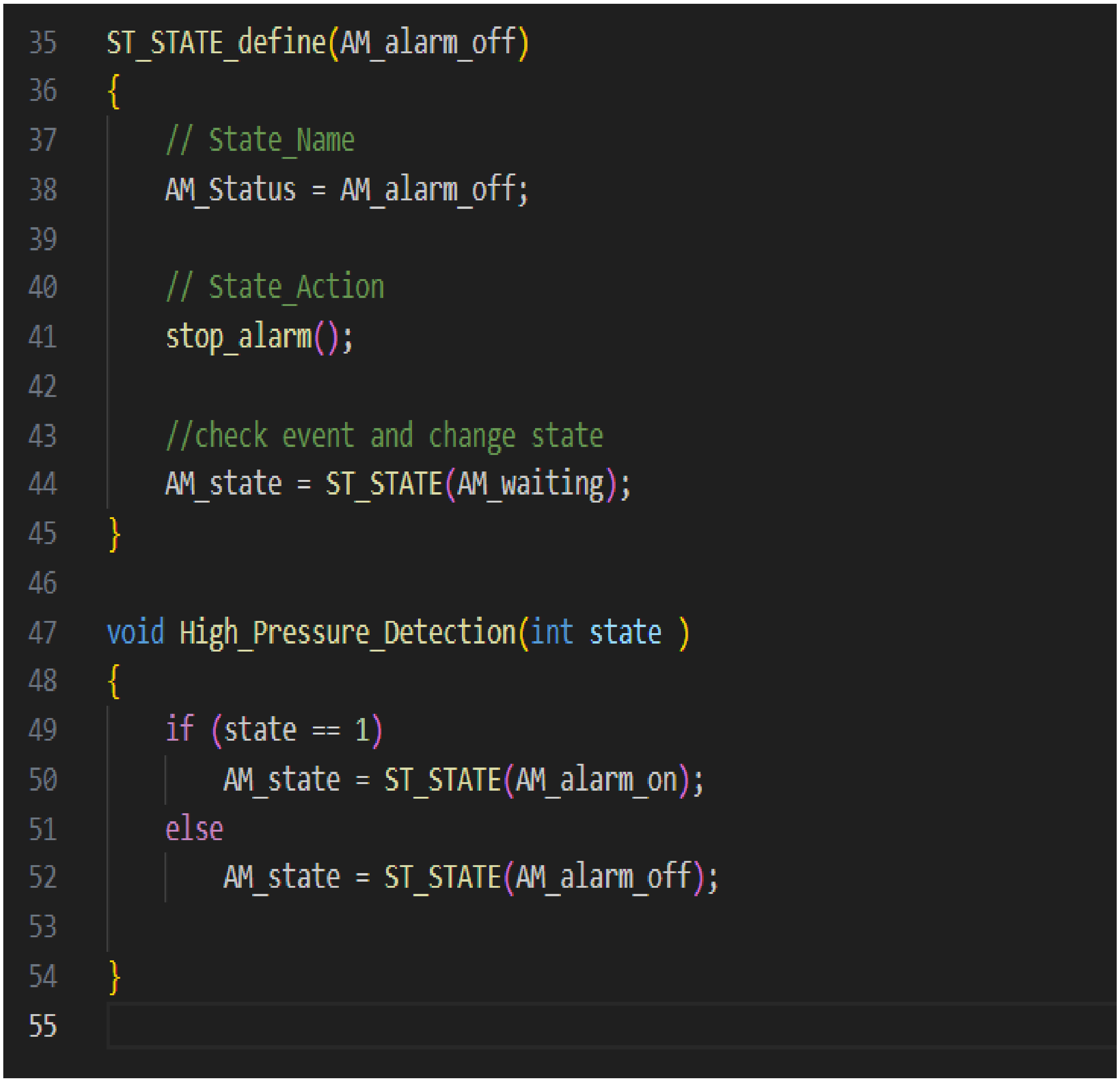
```c
/*
 * Main_Algorithm.h
 *
 *  Created on: Nov 16, 2023
 *      Author: roaa aiman

 */

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

// Define Status
enum {
    MA_waiting,
    MA_High_Pressure
}MA_Status;

// Declare Status Functions CA
ST_STATE_define(MA_waiting);
ST_STATE_define(MA_High_Pressure);

// STATE Pointer to Function
extern void (*MA_state)();


#endif /* MAIN_ALGORITHM_H_ */
```

Alarm Monitor

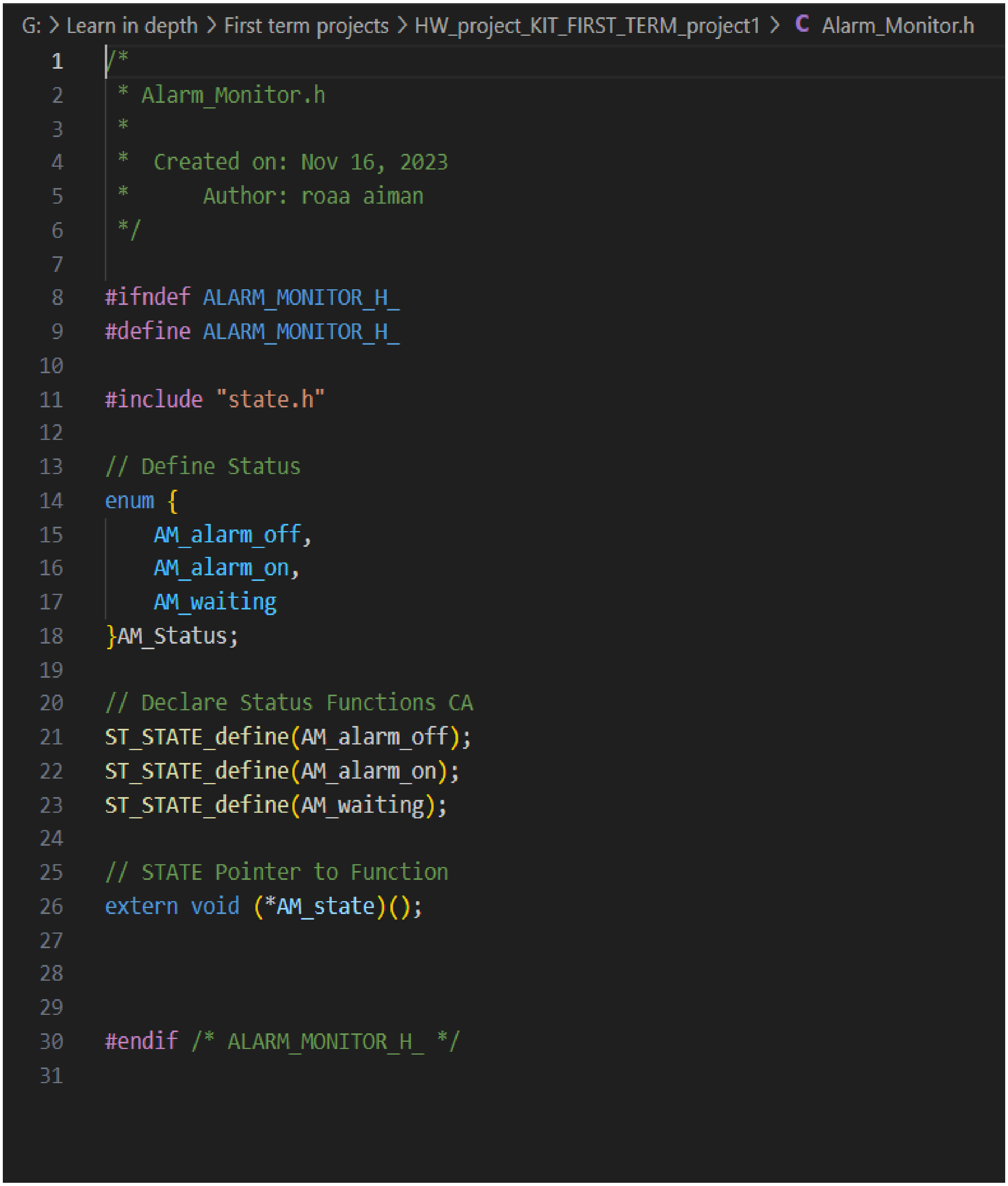


G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > C Alarm_Monitor.c

```
 *      Author: roaa aiman
 */

#include "Alarm_Monitor.h"
// STATE Pointer to function
void (*AM_state)();

// STATE Functions
ST_STATE_define(AM_waiting)
{
    //state_Name
    AM_Status = AM_waiting;
    //timer for time duration 60s
    Delay(60);
    //check event and change state
    AM_state = ST_STATE(AM_alarm_off);
}

ST_STATE_define(AM_alarm_on)
{
    // State_Name
    AM_Status = AM_alarm_on;

    // State_Action
    start_alarm();

    //check event and change state
    AM_state = ST_STATE(AM_waiting);
}
```

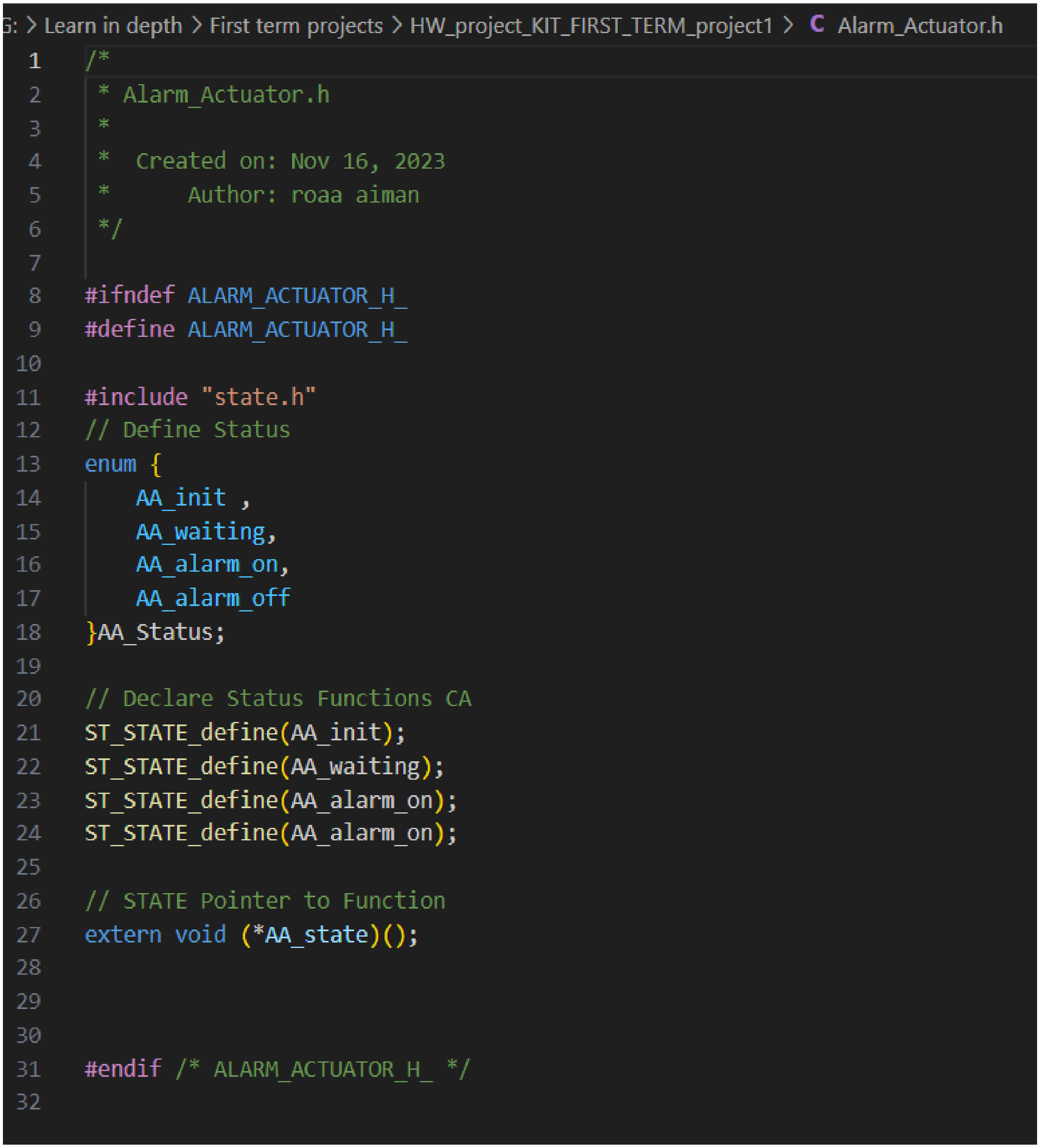


```c
ST_STATE_define(AM_alarm_off)
{
    // State_Name
    AM_Status = AM_alarm_off;

    // State_Action
    stop_alarm();

    //check event and change state
    AM_state = ST_STATE(AM_waiting);
}

void High_Pressure_Detection(int state )
{
    if (state == 1)
        AM_state = ST_STATE(AM_alarm_on);
    else
        AM_state = ST_STATE(AM_alarm_off);

}

```


G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > C Alarm_Monitor.h

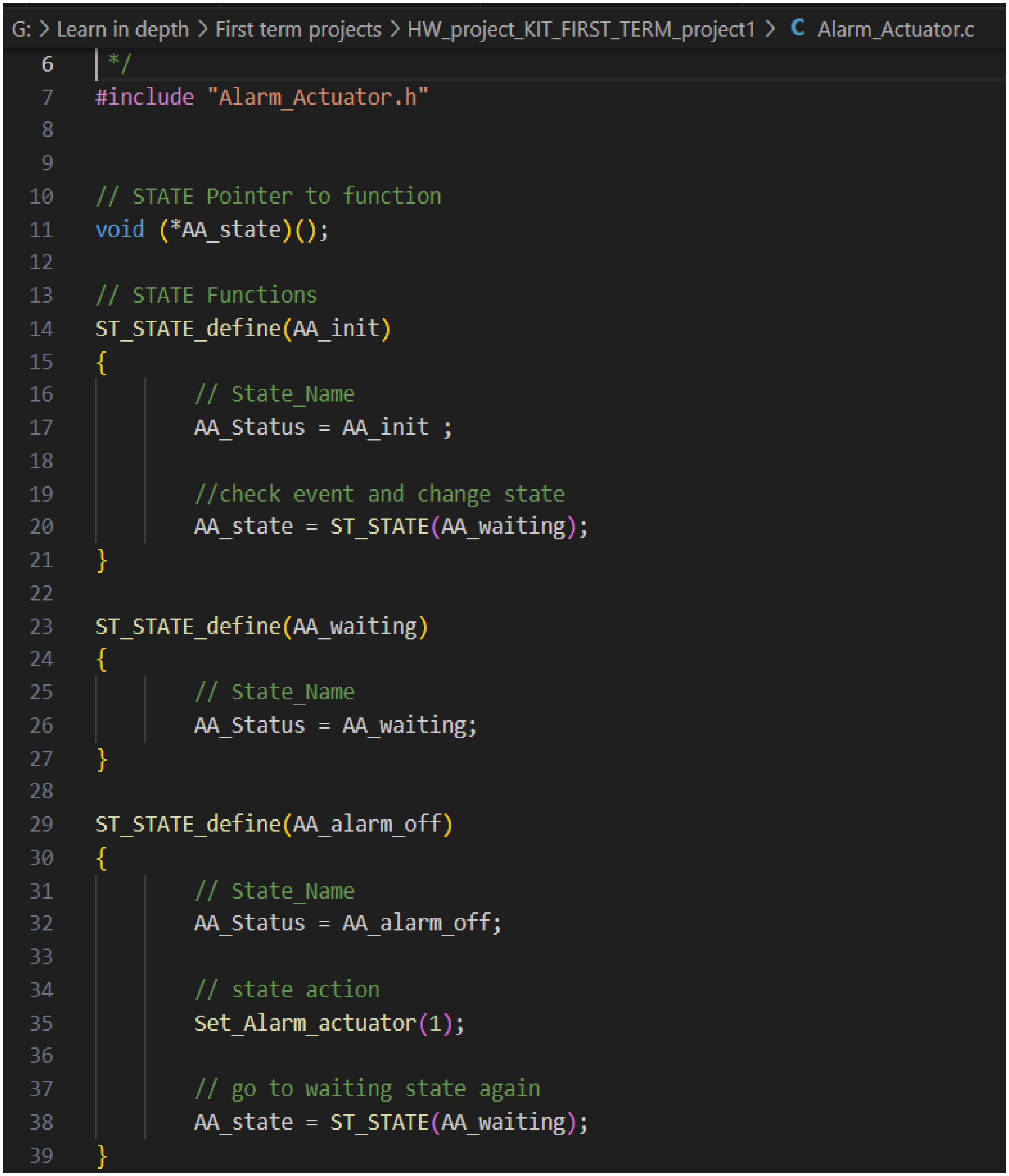
```c
/*
 * Alarm_Monitor.h
 *
 *  Created on: Nov 16, 2023
 *      Author: roaa aiman
 */

#ifndef ALARM_MONITOR_H_
#define ALARM_MONITOR_H_

#include "state.h"

// Define Status
enum {
    AM_alarm_off,
    AM_alarm_on,
    AM_waiting
}AM_Status;

// Declare Status Functions CA
ST_STATE_define(AM_alarm_off);
ST_STATE_define(AM_alarm_on);
ST_STATE_define(AM_waiting);

// STATE Pointer to Function
extern void (*AM_state)();



#endif /* ALARM_MONITOR_H_ */
```

Alarm Actuator



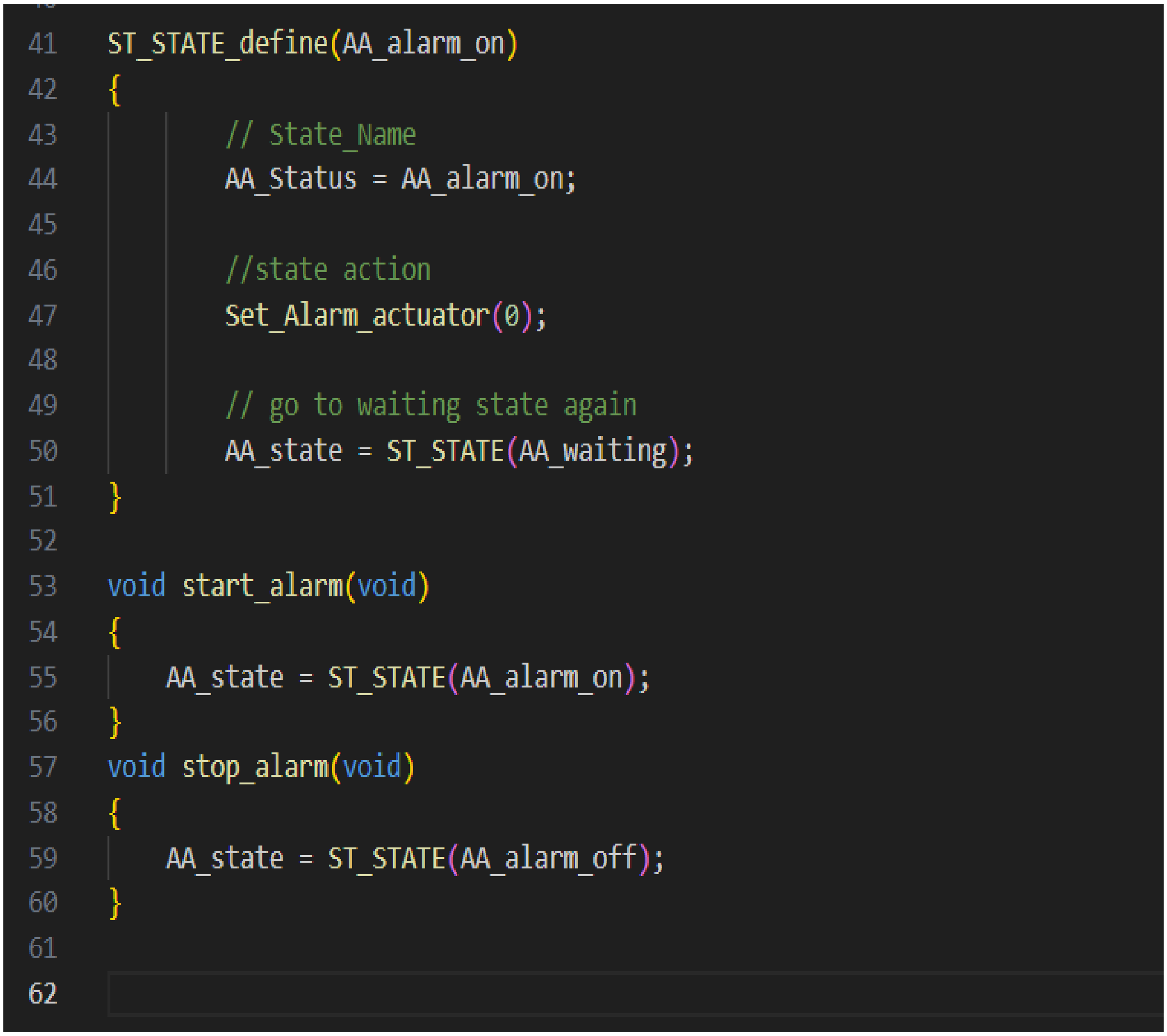
G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > C Alarm_Actuator.h

```c
/*
 * Alarm_Actuator.h
 *
 *  Created on: Nov 16, 2023
 *      Author: roaa aiman
 */

#ifndef ALARM_ACTUATOR_H_
#define ALARM_ACTUATOR_H_

#include "state.h"
// Define Status
enum {
    AA_init ,
    AA_waiting,
    AA_alarm_on,
    AA_alarm_off
}AA_Status;

// Declare Status Functions CA
ST_STATE_define(AA_init);
ST_STATE_define(AA_waiting);
ST_STATE_define(AA_alarm_on);
ST_STATE_define(AA_alarm_on);

// STATE Pointer to Function
extern void (*AA_state)();



#endif /* ALARM_ACTUATOR_H_ */

```



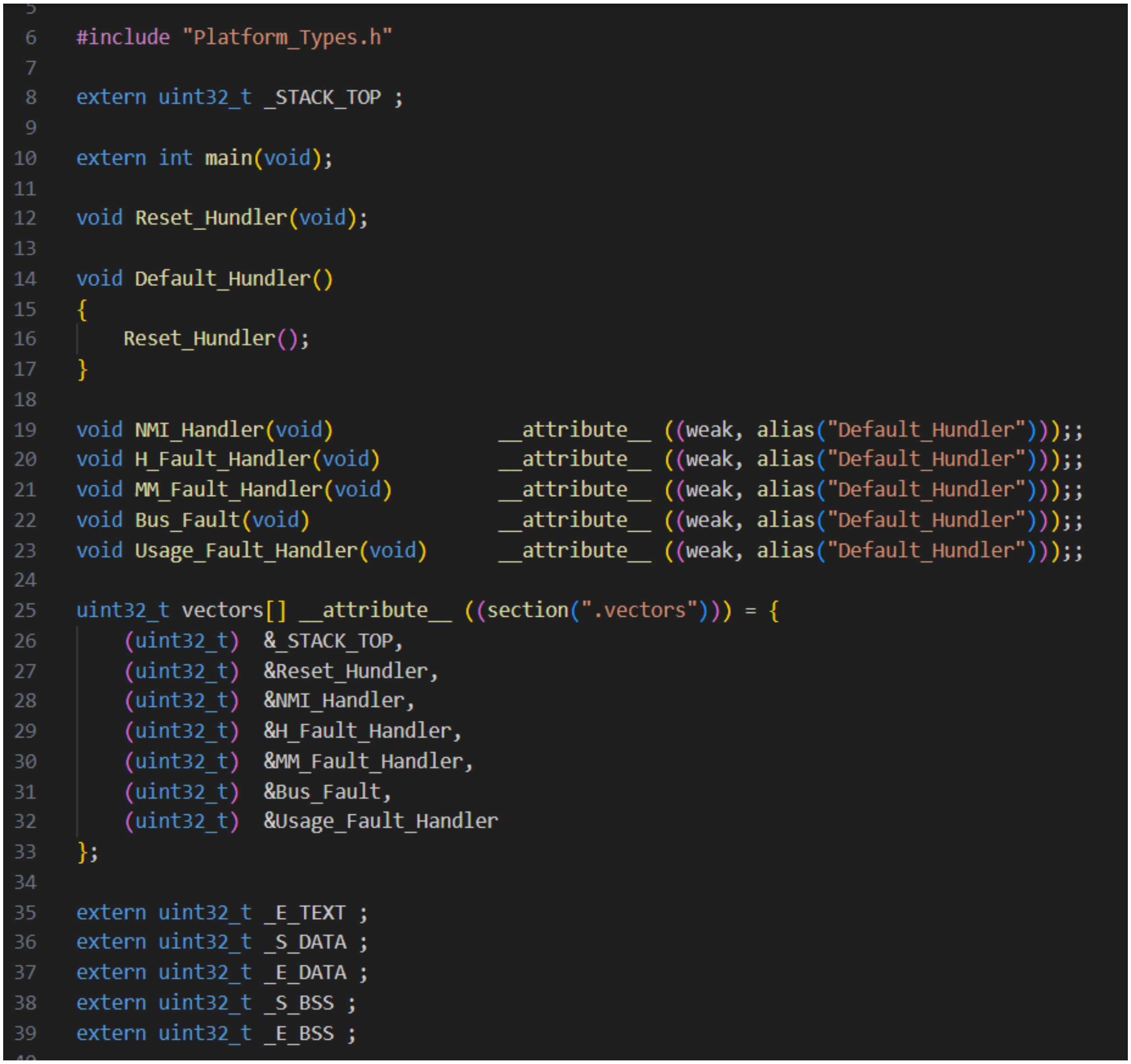
G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > C Alarm_Actuator.c

```c
 */
#include "Alarm_Actuator.h"


// STATE Pointer to function
void (*AA_state)();

// STATE Functions
ST_STATE_define(AA_init)
{
        // State_Name
        AA_Status = AA_init ;

        //check event and change state
        AA_state = ST_STATE(AA_waiting);
}

ST_STATE_define(AA_waiting)
{
        // State_Name
        AA_Status = AA_waiting;
}

ST_STATE_define(AA_alarm_off)
{
        // State_Name
        AA_Status = AA_alarm_off;

        // state action
        Set_Alarm_actuator(1);

        // go to waiting state again
        AA_state = ST_STATE(AA_waiting);
}
```

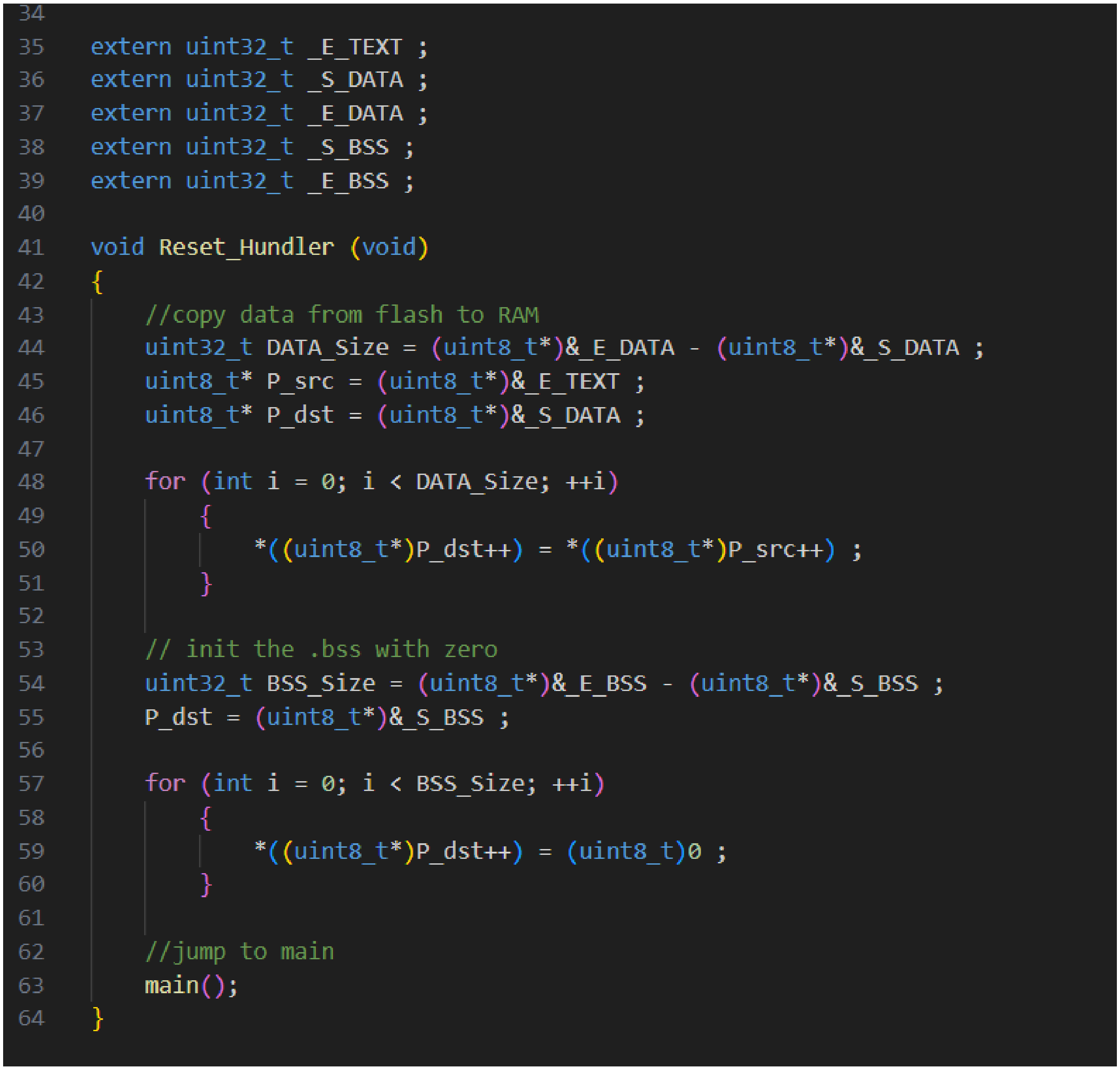


```
ST_STATE_define(AA_alarm_on)
{
        // State_Name
        AA_Status = AA_alarm_on;

        //state action
        Set_Alarm_actuator(0);

        // go to waiting state again
        AA_state = ST_STATE(AA_waiting);
}

void start_alarm(void)
{
    AA_state = ST_STATE(AA_alarm_on);
}
void stop_alarm(void)
{
    AA_state = ST_STATE(AA_alarm_off);
}

```

Startup.c

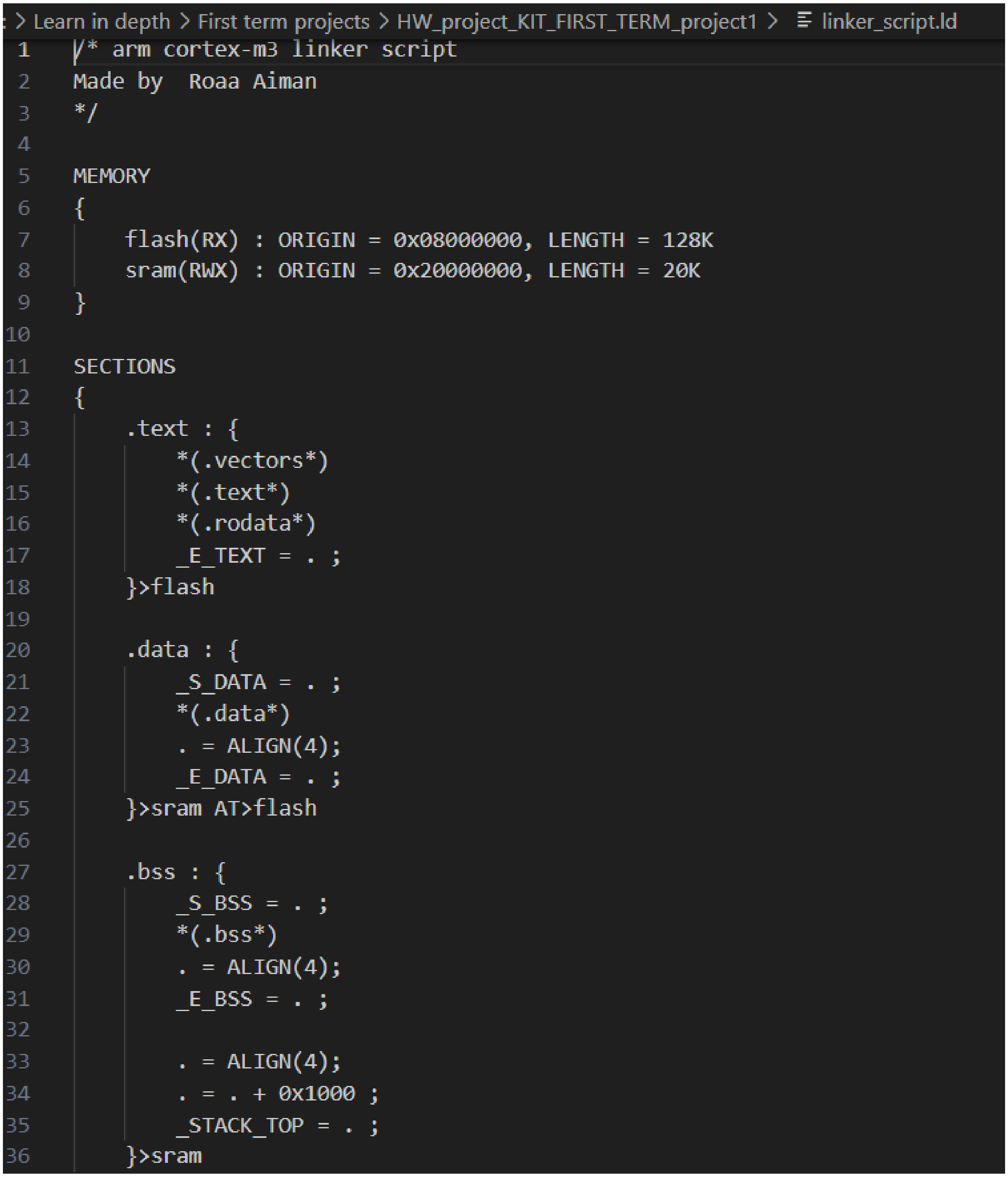


```c
#include "Platform_Types.h"

extern uint32_t _STACK_TOP ;

extern int main(void);

void Reset_Hundler(void);

void Default_Hundler()
{
    Reset_Hundler();
}

void NMI_Handler(void)              __attribute__ ((weak, alias("Default_Hundler")));;
void H_Fault_Handler(void)          __attribute__ ((weak, alias("Default_Hundler")));;
void MM_Fault_Handler(void)         __attribute__ ((weak, alias("Default_Hundler")));;
void Bus_Fault(void)                __attribute__ ((weak, alias("Default_Hundler")));;
void Usage_Fault_Handler(void)      __attribute__ ((weak, alias("Default_Hundler")));;

uint32_t vectors[] __attribute__ ((section(".vectors"))) = {
    (uint32_t)  &_STACK_TOP,
    (uint32_t)  &Reset_Hundler,
    (uint32_t)  &NMI_Handler,
    (uint32_t)  &H_Fault_Handler,
    (uint32_t)  &MM_Fault_Handler,
    (uint32_t)  &Bus_Fault,
    (uint32_t)  &Usage_Fault_Handler
};

extern uint32_t _E_TEXT ;
extern uint32_t _S_DATA ;
extern uint32_t _E_DATA ;
extern uint32_t _S_BSS ;
extern uint32_t _E_BSS ;
```



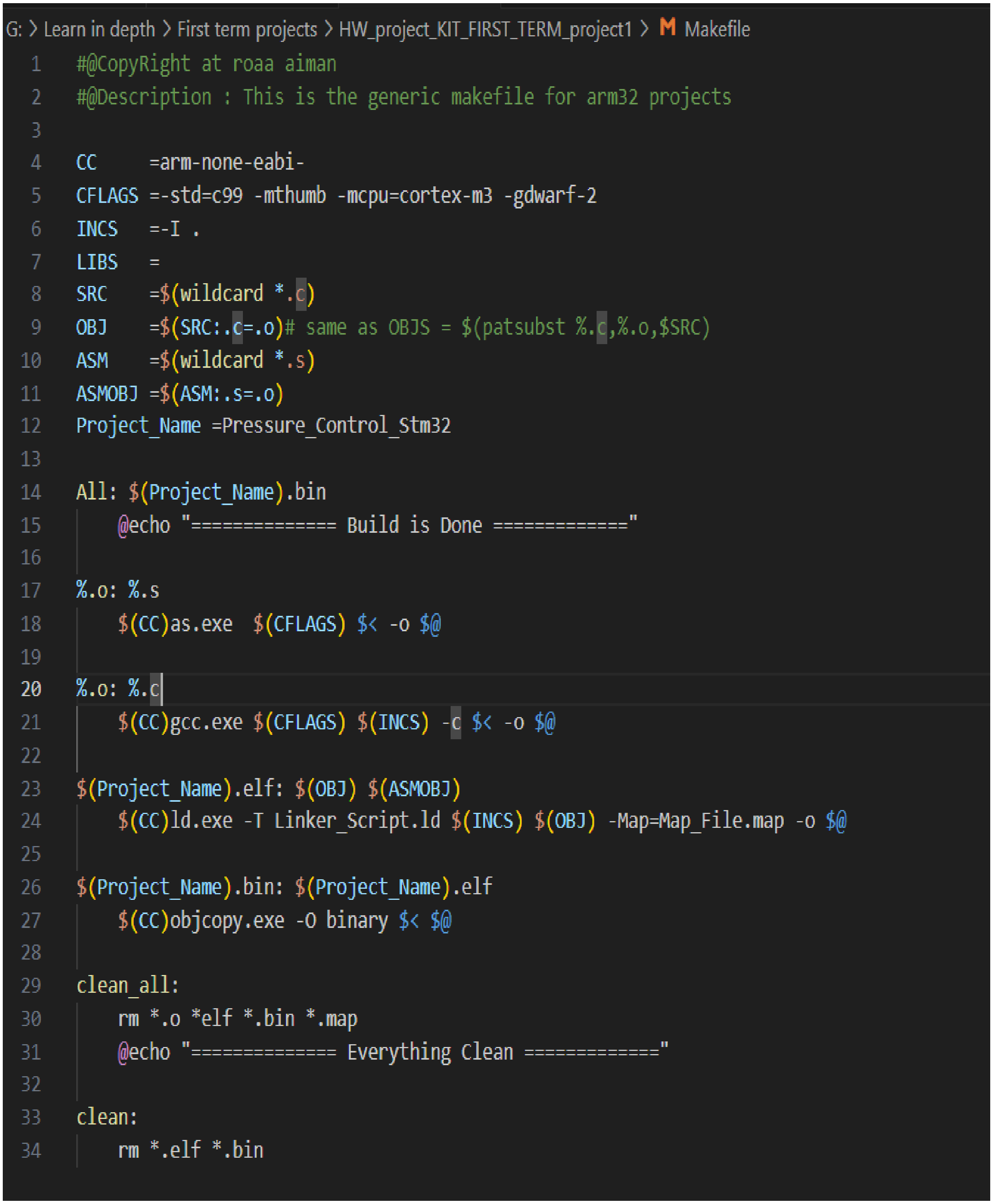
```c
extern uint32_t _E_TEXT ;
extern uint32_t _S_DATA ;
extern uint32_t _E_DATA ;
extern uint32_t _S_BSS ;
extern uint32_t _E_BSS ;

void Reset_Hundler (void)
{
    //copy data from flash to RAM
    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 the .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 ;
        }

    //jump to main
    main();
}
```

Linker_script.ld


: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > linker_script.ld

```
/* arm cortex-m3 linker script
Made by  Roaa Aiman
*/

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*)
        . = ALIGN(4);
        _E_DATA = . ;
    }>sram AT>flash

    .bss : {
        _S_BSS = . ;
        *(.bss*)
        . = ALIGN(4);
        _E_BSS = . ;

        . = ALIGN(4);
        . = . + 0x1000 ;
        _STACK_TOP = . ;
    }>sram
```

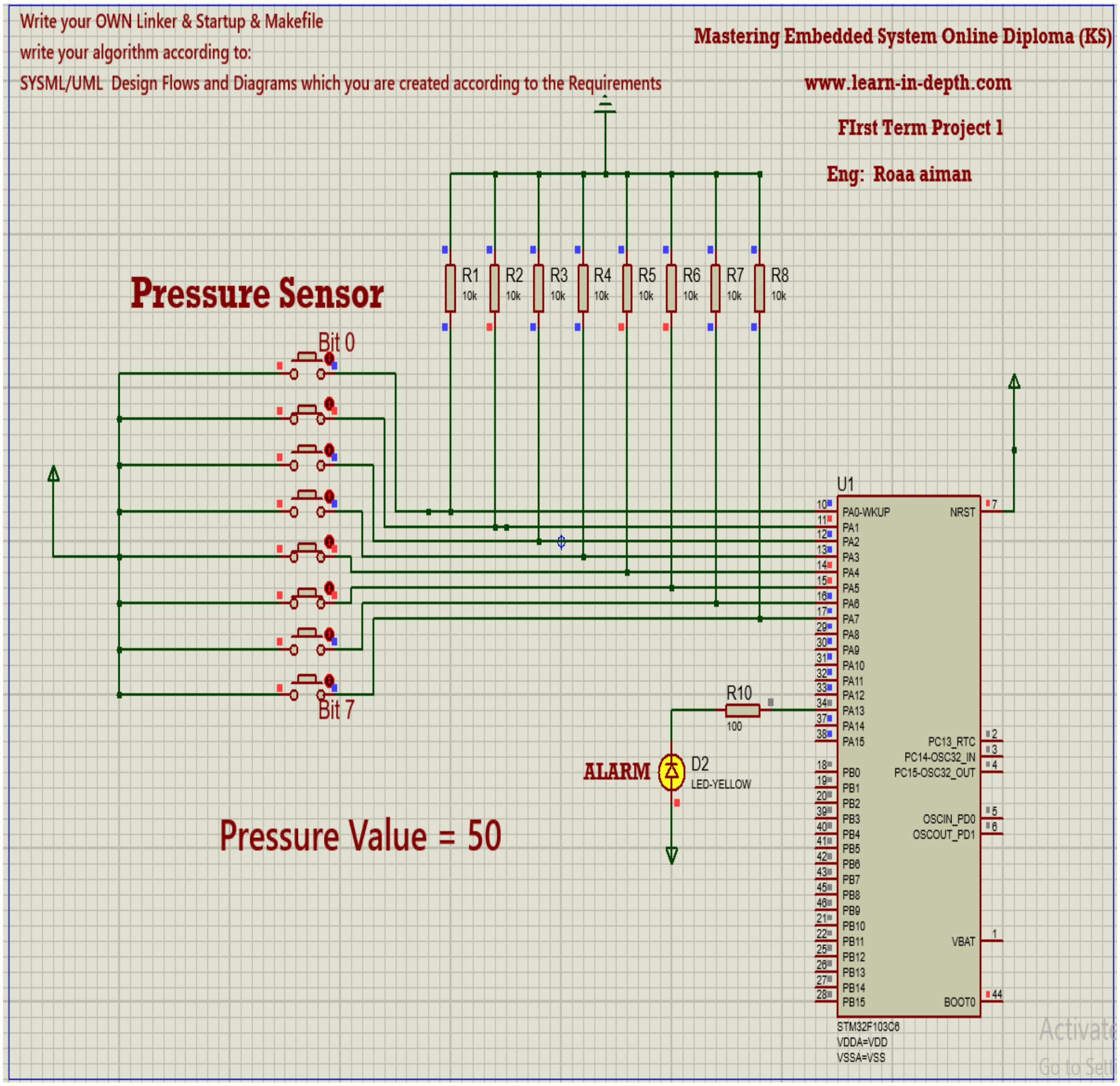
Make File



G: > Learn in depth > First term projects > HW_project_KIT_FIRST_TERM_project1 > M Makefile

```makefile
#@CopyRight at roaa aiman
#@Description : This is the generic makefile for arm32 projects

CC     =arm-none-eabi-
CFLAGS =-std=c99 -mthumb -mcpu=cortex-m3 -gdwarf-2
INCS   =-I .
LIBS   =
SRC    =$(wildcard *.c)
OBJ    =$(SRC:.c=.o)# same as OBJS = $(patsubst %.c,%.o,$SRC)
ASM    =$(wildcard *.s)
ASMOBJ =$(ASM:.s=.o)
Project_Name =Pressure_Control_Stm32

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

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

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

$(Project_Name).elf: $(OBJ) $(ASMOBJ)
	$(CC)ld.exe -T Linker_Script.ld $(INCS) $(OBJ) -Map=Map_File.map -o $@

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

clean_all:
	rm *.o *elf *.bin *.map
	@echo "============= Everything Clean ============"

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

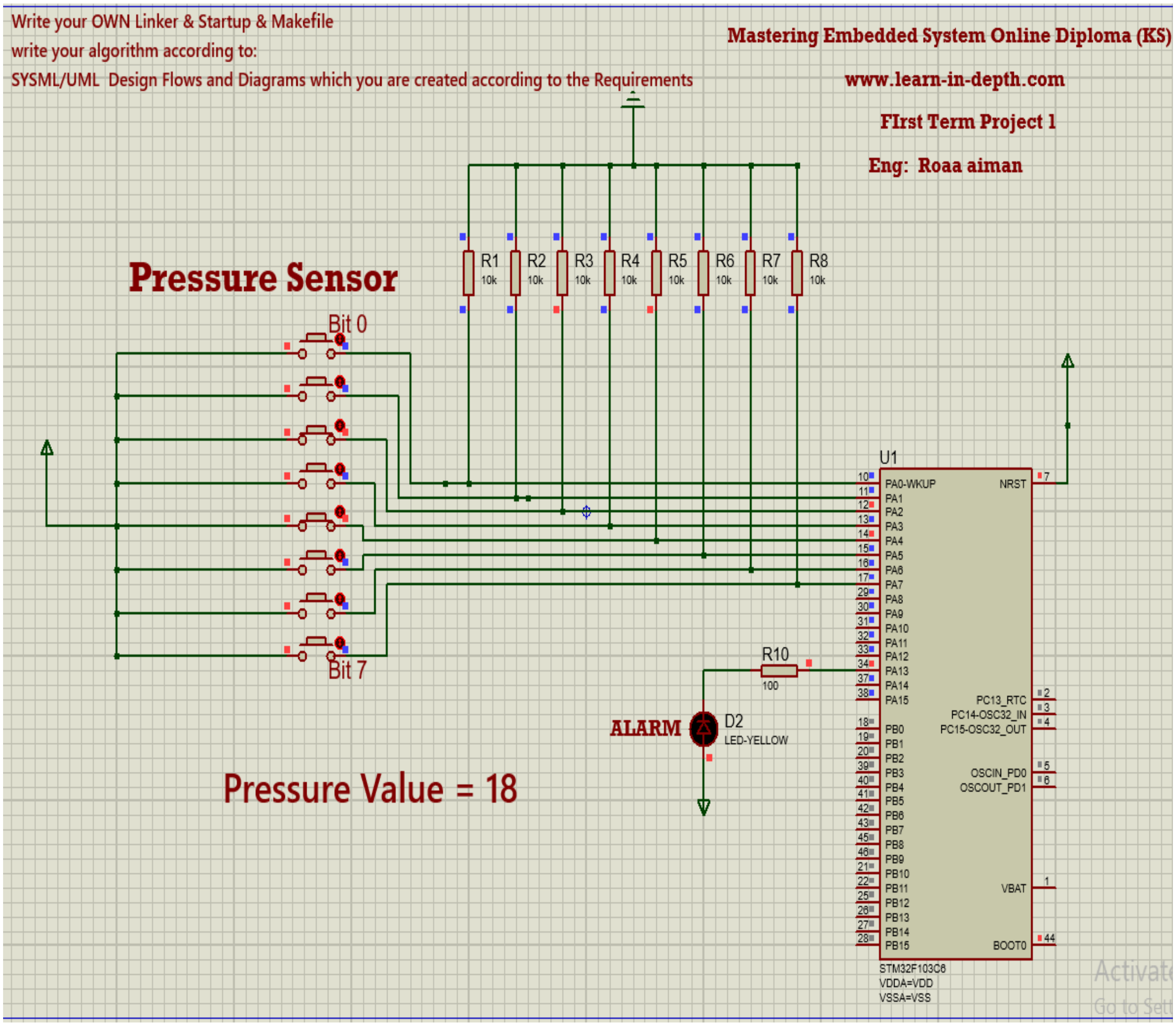
Output of the program



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: Roaa aiman

Pressure Value = 50



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: Roaa aiman