Mastering Embedded System Online Diploma

www.learn-in-depth.com

Pressure Controller Project Report

First Term (Final Project 1)

Eng. Abdelaziz Maher Abdelaziz

My Profile:

www.learn-in-depth.com/online-diploma/abdelazizmaher17499@gmail.com

Case study:

A” client” expects you to deliver the software of the following system:

- **Specification** (from the client):
  - 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.
  - keeps track of the measured values.

Pressure Controller Assumptions:

- The controller setup 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 a power cut

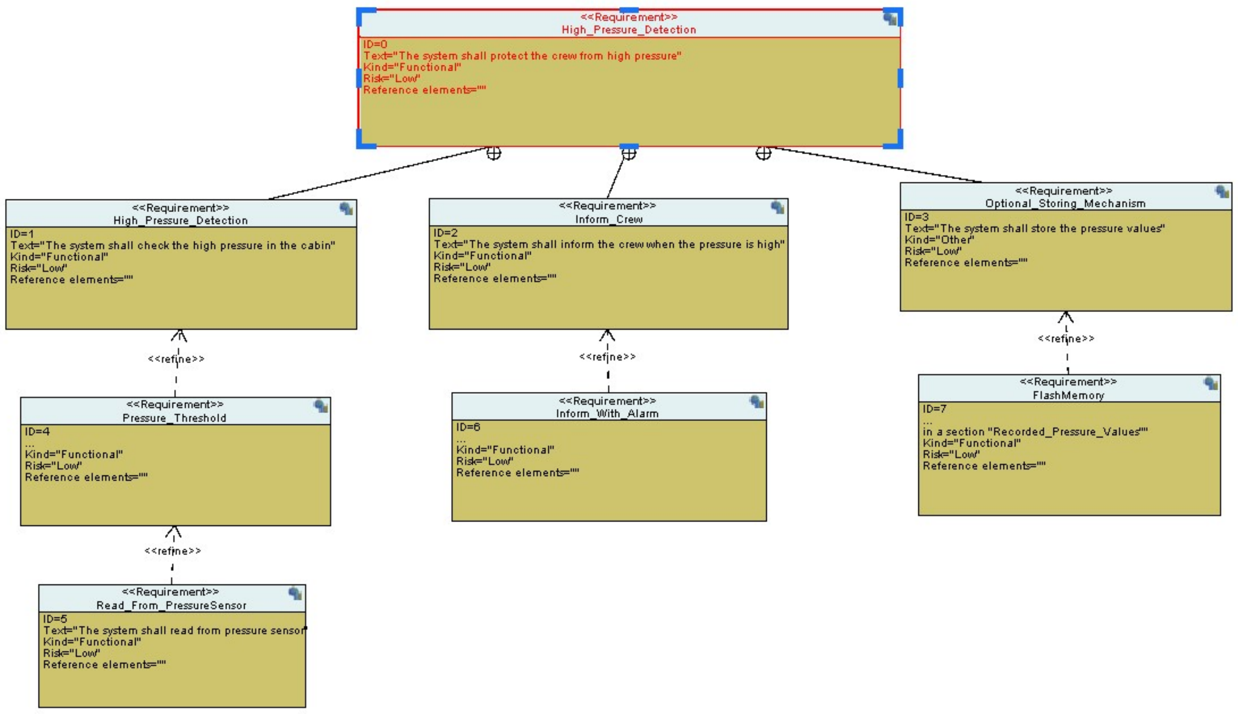
Versioning: The” keep track of measured value” option is not modeled in the first version of the design.

Method:

Verification and Validation (V&V) model is used in this project.

Requirements:

System Requirement Diagram:



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

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

<<Requirement>>
Inform_Crew
ID=2
Text="The system shall inform the crew when the pressure is high"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Optional_Storing_Mechanism
ID=3
Text="The system shall store the pressure values"
Kind="Other"
Risk="Low"
Reference elements=""

<<Requirement>>
Pressure_Threshold
ID=4
...
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Inform_With_Alarm
ID=6
...
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
FlashMemory
ID=7
...
in a section "Recorded_Pressure_Values""
Kind="Functional"
Risk="Low"
Reference elements=""

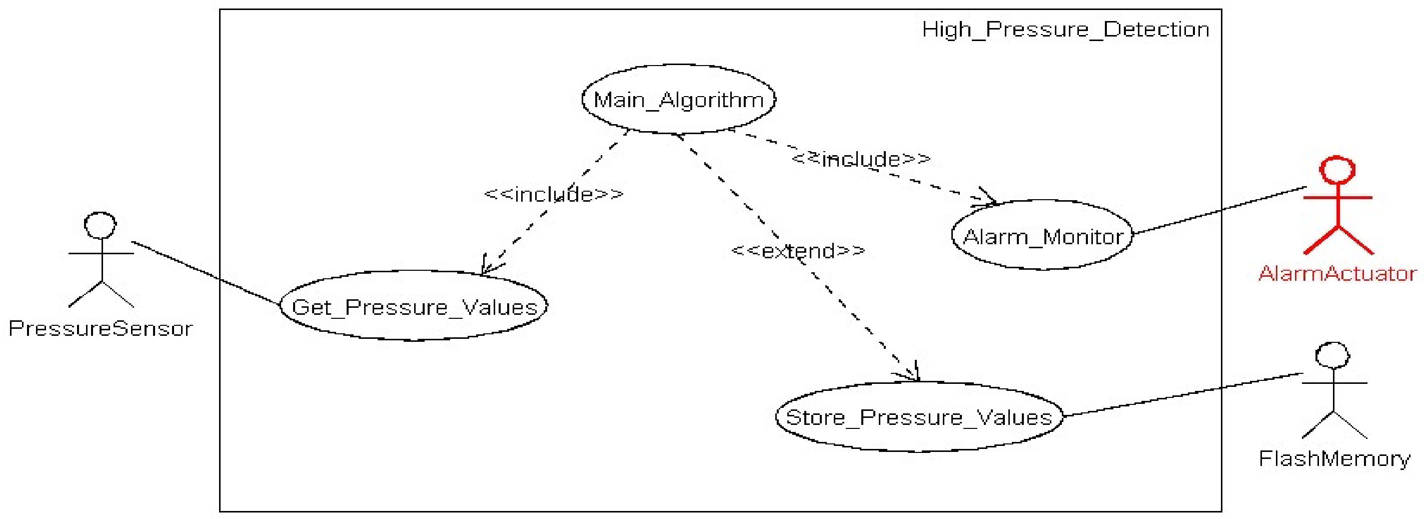
<<Requirement>>
Read_From_PressureSensor
ID=5
Text="The system shall read from pressure sensor"
Kind="Functional"
Risk="Low"
Reference elements=""

Space Exploration:

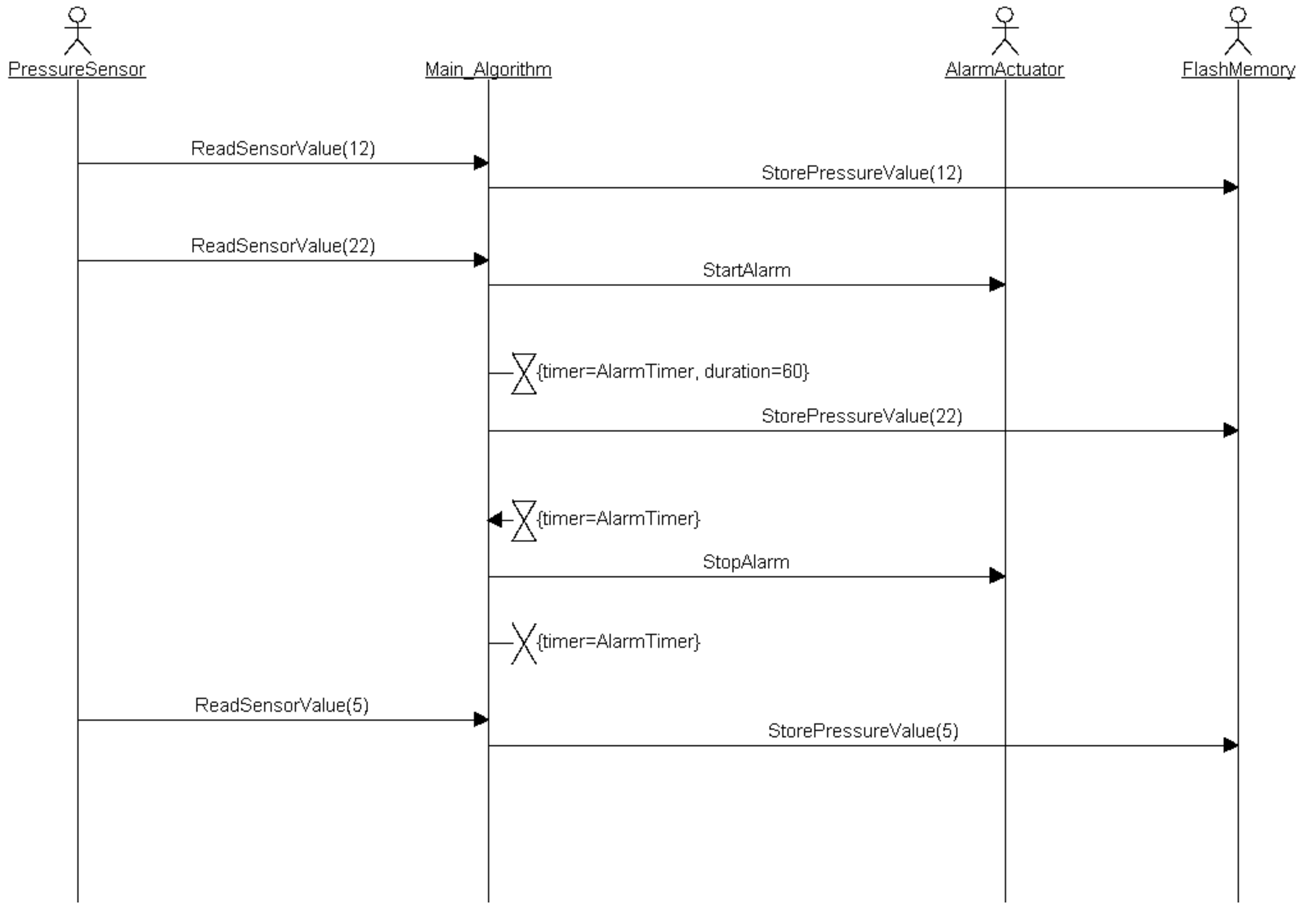
Cortex-m3 STM32F103C6 microcontroller is chosen.

System Analysis:

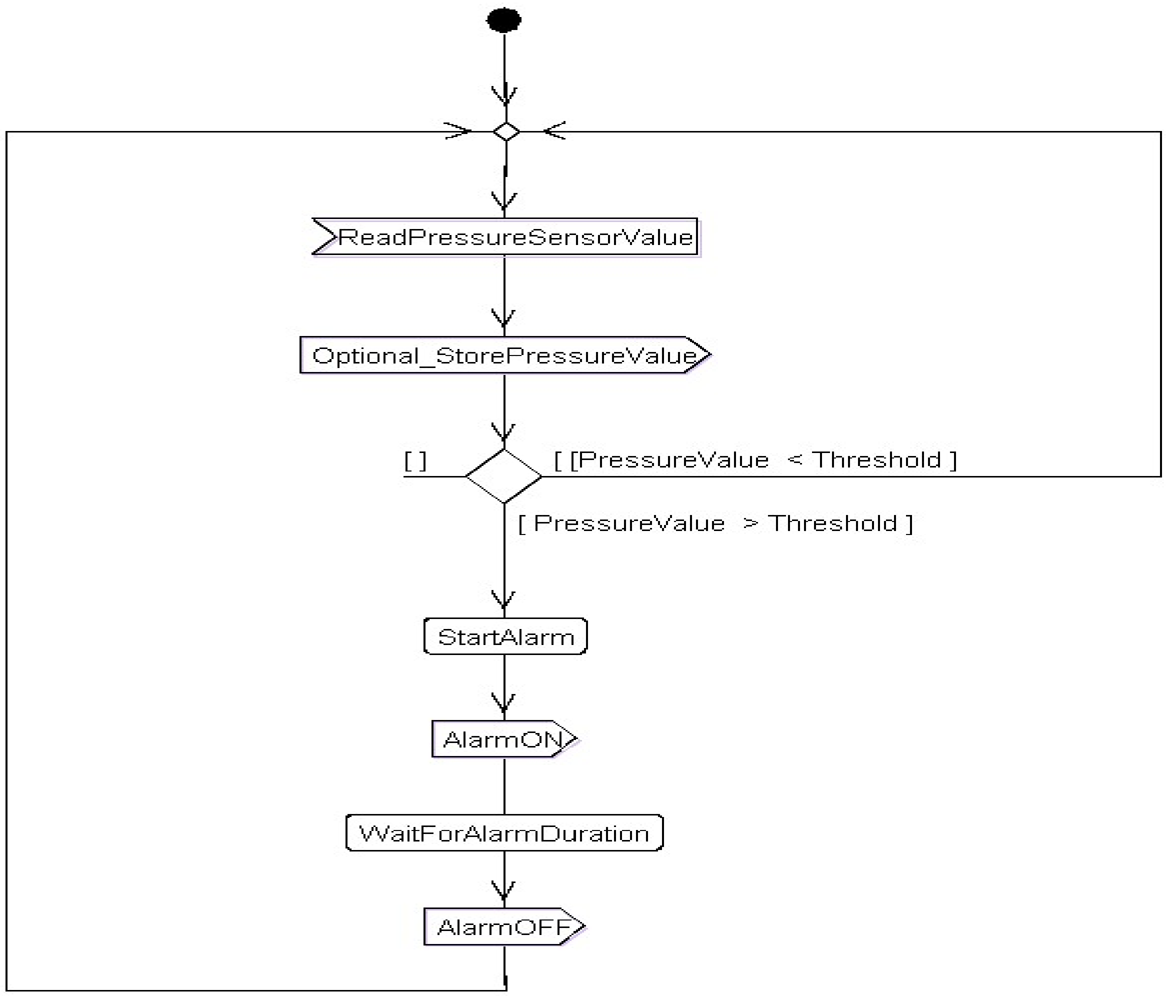
- Use case diagram:



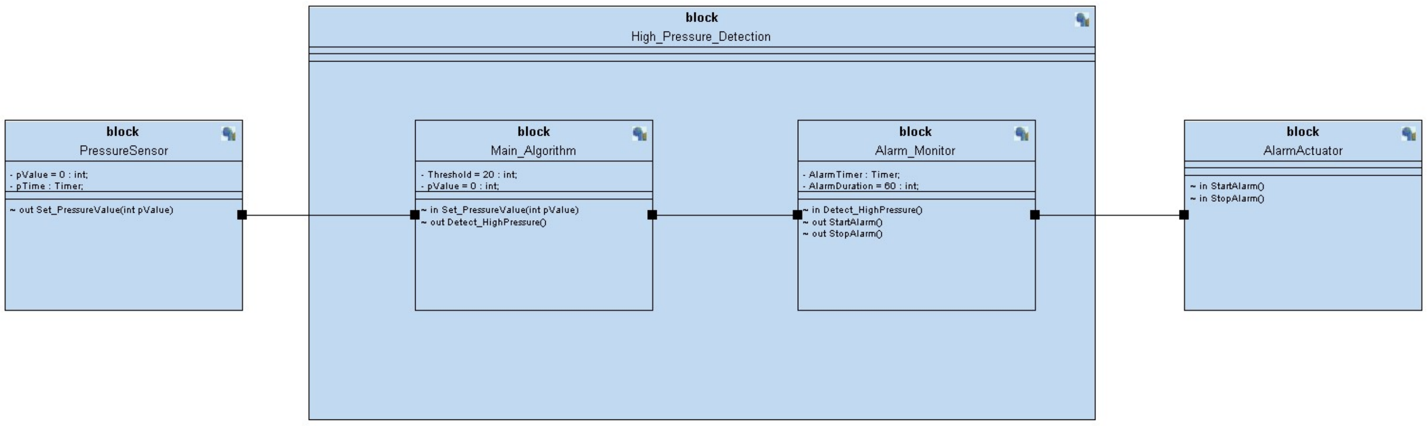
- Sequence diagram:



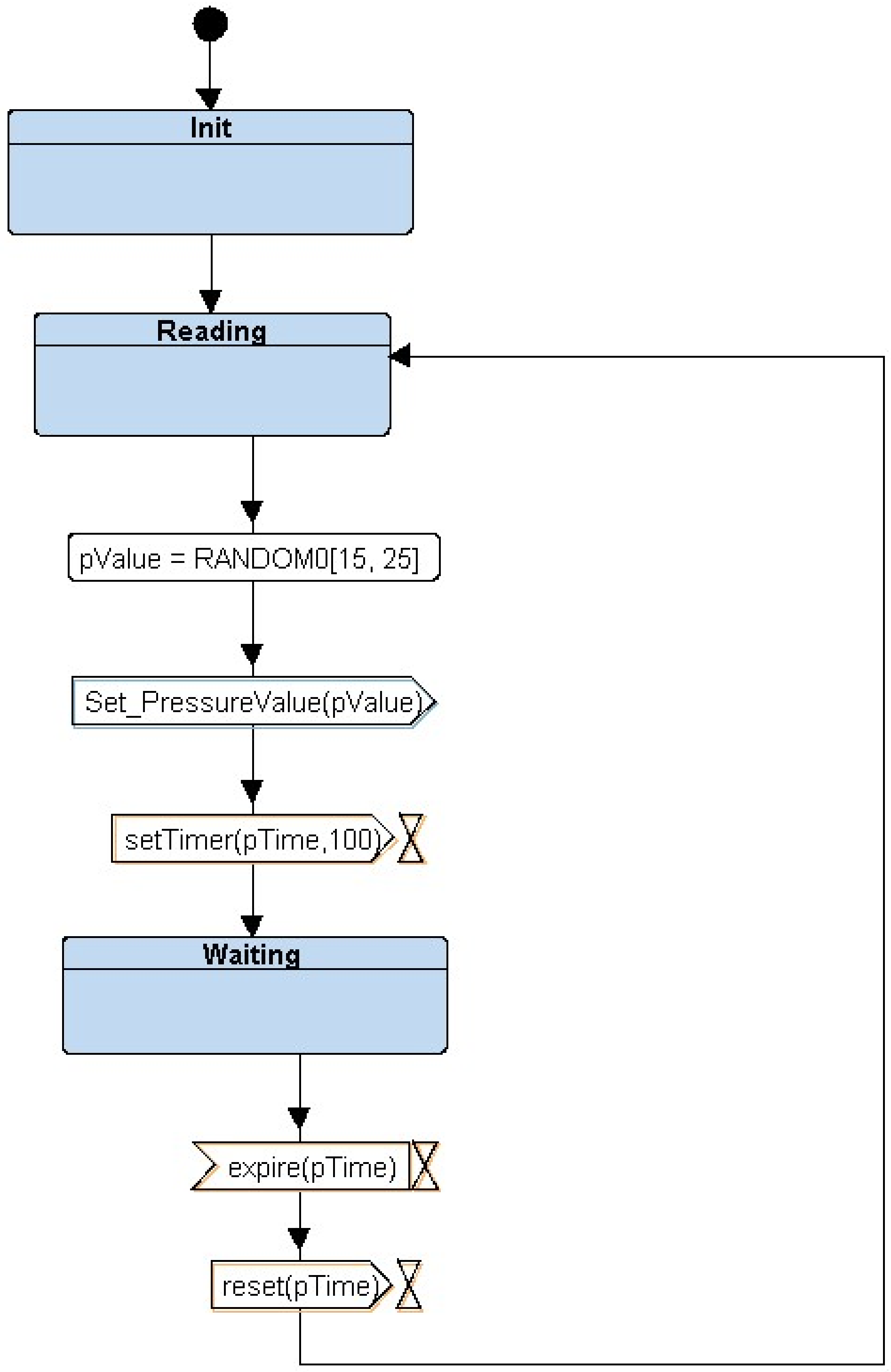
- Activity diagram:


ReadPressureSensorValue
Optional_StorePressureValue
[ ]
[[PressureValue < Threshold]
[PressureValue > Threshold]
StartAlarm
AlarmON
WaitForAlarmDuration
AlarmOFF

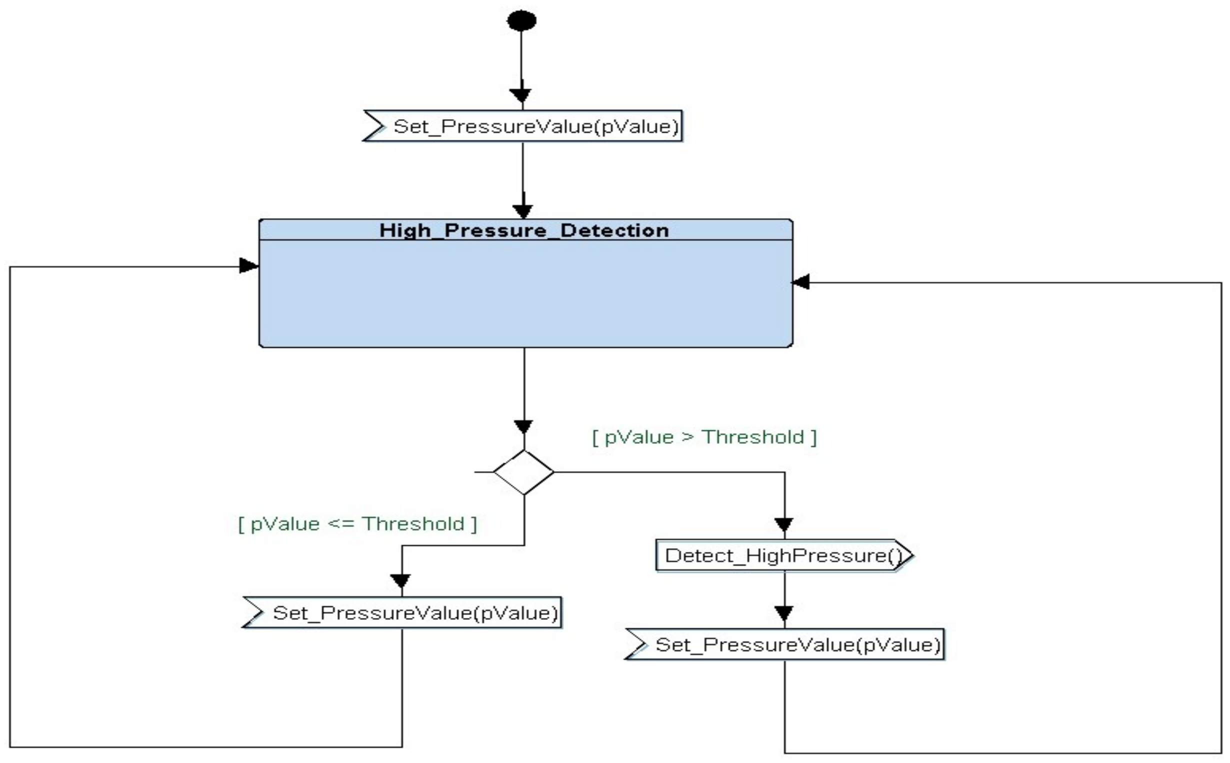
System design:



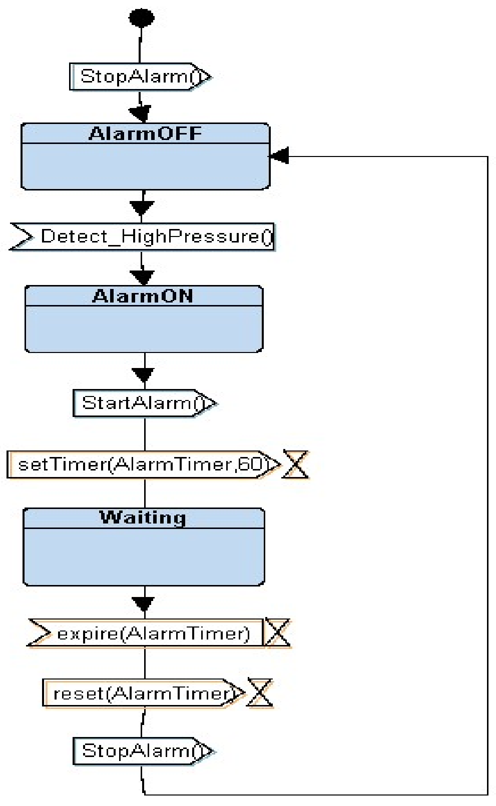
1) Pressure Sensor State Diagram:



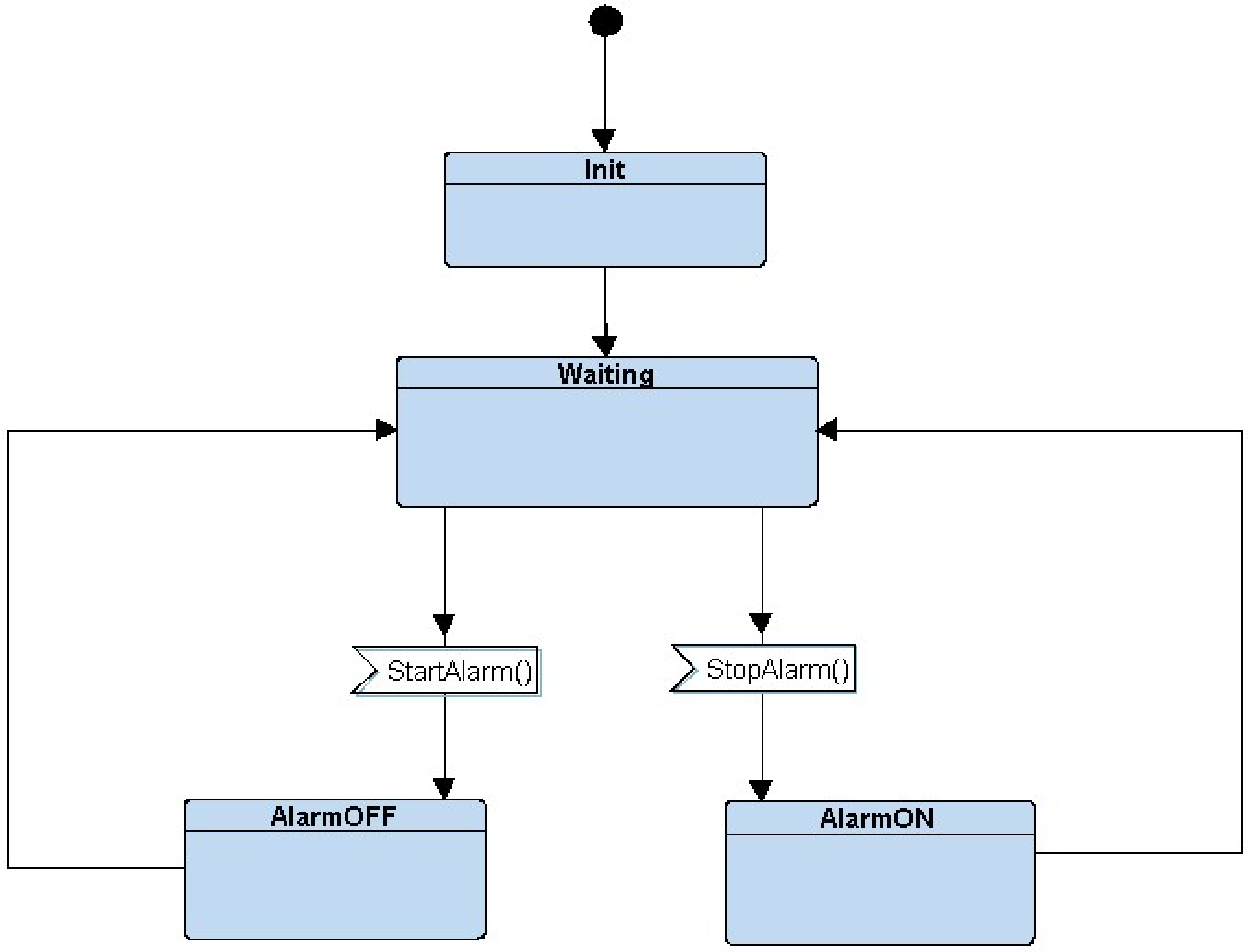
2) Main Algorithm state diagram:



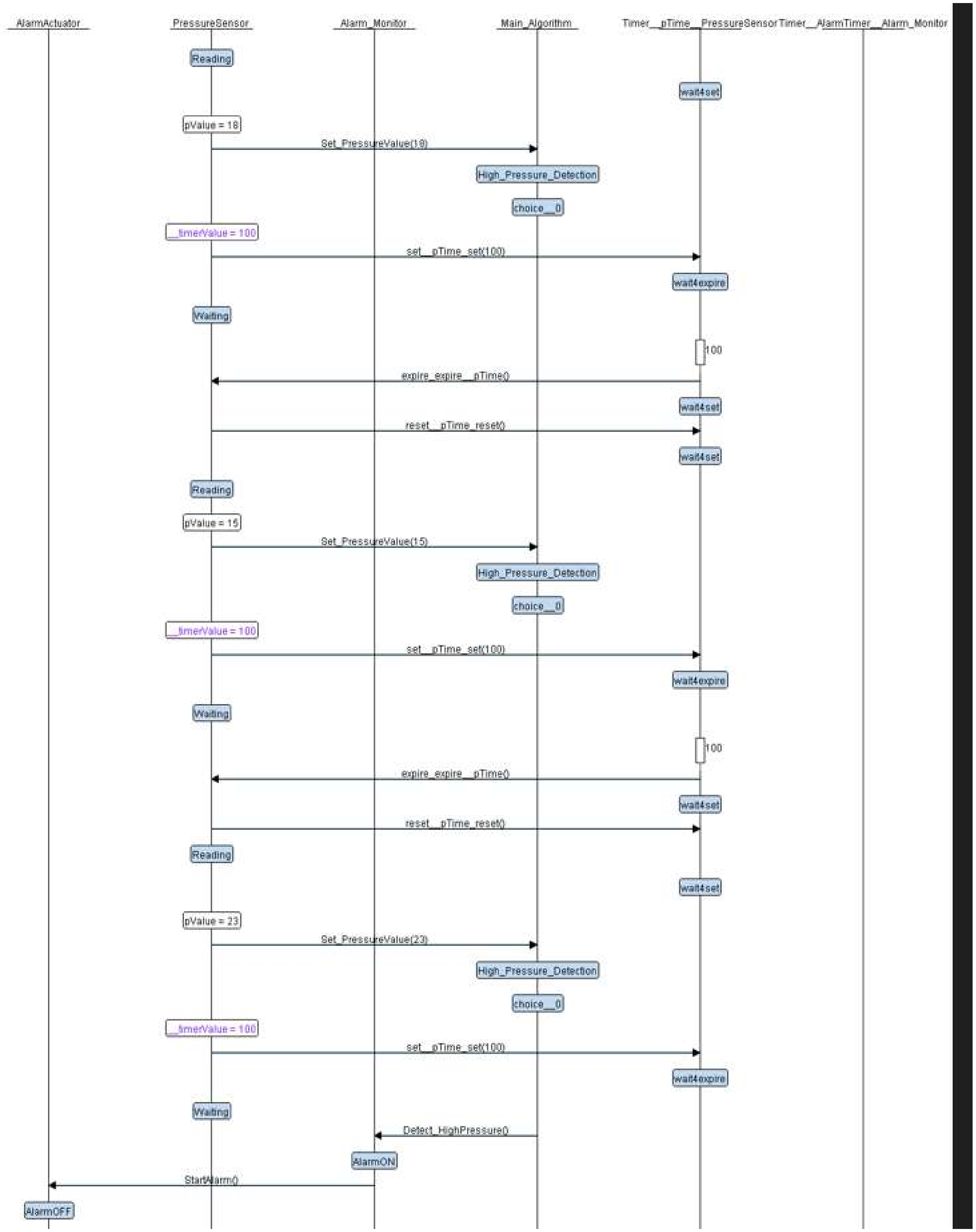
3) Alarm Monitor state diagram:



4) Alarm actuator state diagram:

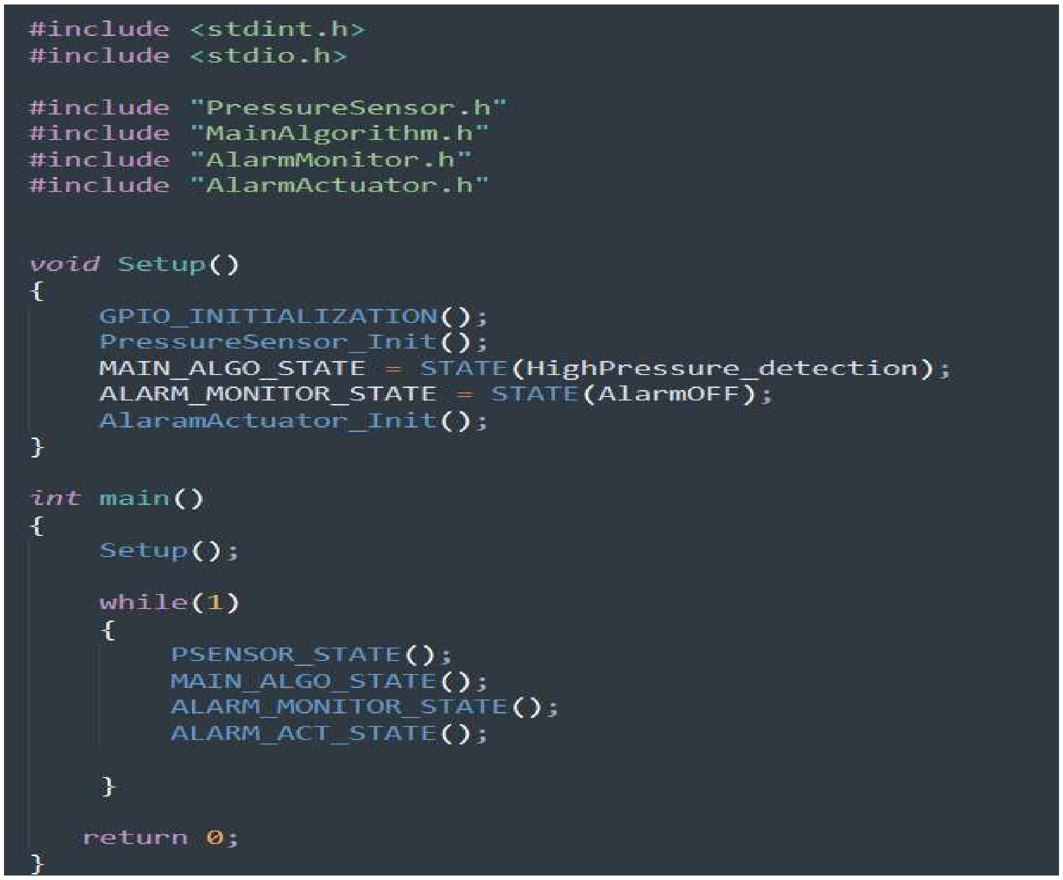


Simulation:

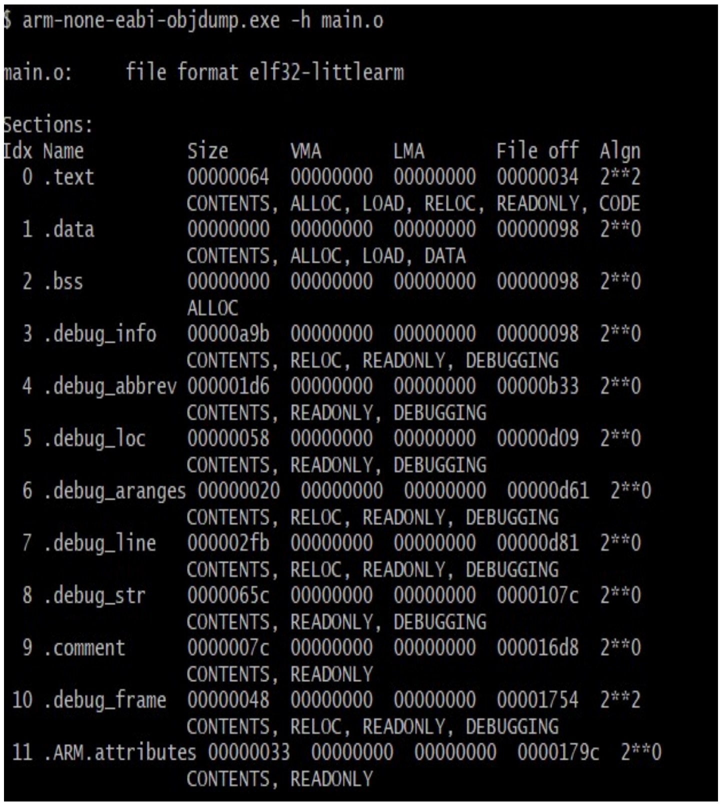

AlarmActuator
PressureSensor
Alarm_Monitor
Main_Algorithm
Timer__pTime__PressureSensor Timer__AlarmTimer__Alarm_Monitor
Reading
wait4set
pValue = 18
Set_PressureValue(18)
High_Pressure_Detection
choice__0
__timerValue = 100
set__pTime_set(100)
wait4expire
Waiting
100
expire_expire__pTime()
wait4set
reset__pTime_reset()
wait4set
Reading
pValue = 15
Set_PressureValue(15)
High_Pressure_Detection
choice__0
__timerValue = 100
set__pTime_set(100)
wait4expire
Waiting
100
expire_expire__pTime()
wait4set
reset__pTime_reset()
Reading
wait4set
pValue = 23
Set_PressureValue(23)
High_Pressure_Detection
choice__0
__timerValue = 100
set__pTime_set(100)
wait4expire
Waiting
Detect_HighPressure()
AlarmON
StartAlarm()
AlarmOFF

Software Implementation:

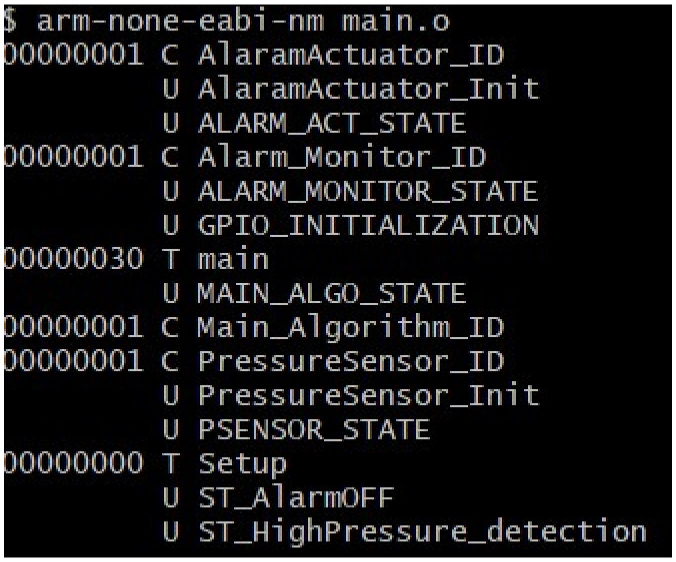
- Main:



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

#include "PressureSensor.h"
#include "MainAlgorithm.h"
#include "AlarmMonitor.h"
#include "AlarmActuator.h"


void Setup()
{
    GPIO_INITIALIZATION();
    PressureSensor_Init();
    MAIN_ALGO_STATE = STATE(HighPressure_detection);
    ALARM_MONITOR_STATE = STATE(AlarmOFF);
    AlaramActuator_Init();
}

int main()
{
    Setup();

    while(1)
    {
        PSENSOR_STATE();
        MAIN_ALGO_STATE();
        ALARM_MONITOR_STATE();
        ALARM_ACT_STATE();

    }

   return 0;
}
```

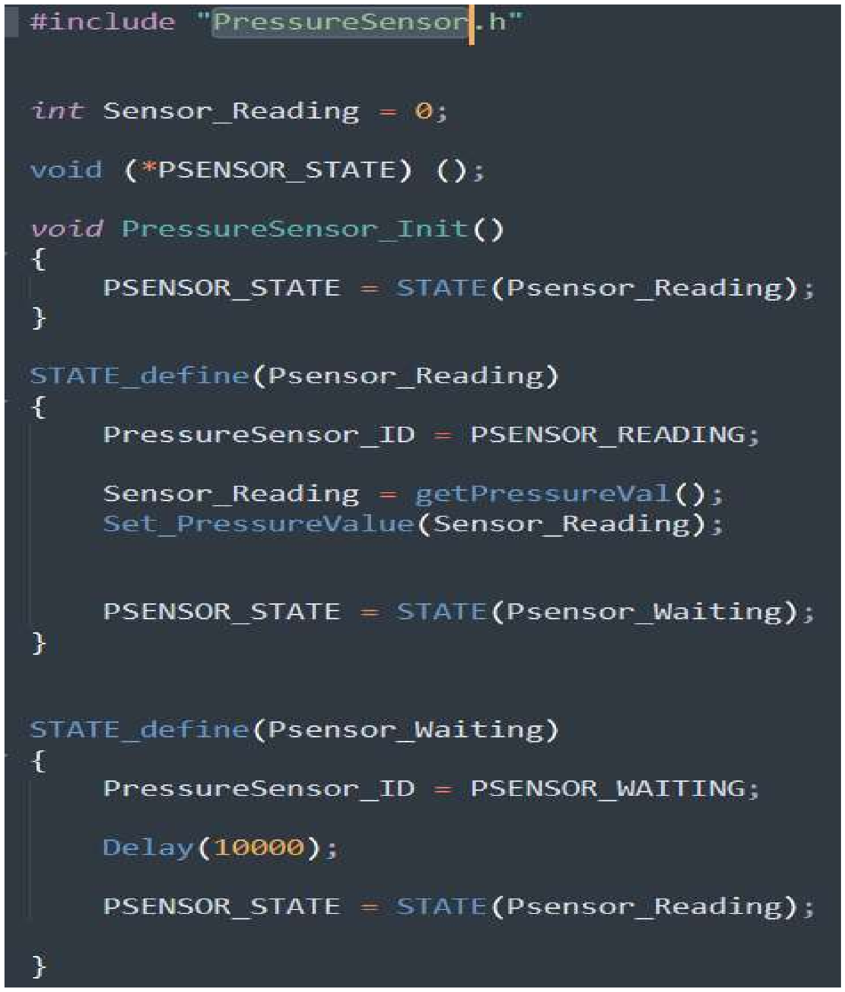


```
$ arm-none-eabi-objdump.exe -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000064  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000098  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000098  2**0
                  ALLOC
  3 .debug_info   00000a9b  00000000  00000000  00000098  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001d6  00000000  00000000  00000b33  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000058  00000000  00000000  00000d09  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000d61  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002fb  00000000  00000000  00000d81  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000065c  00000000  00000000  0000107c  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  000016d8  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000048  00000000  00000000  00001754  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  0000179c  2**0
                  CONTENTS, READONLY
```

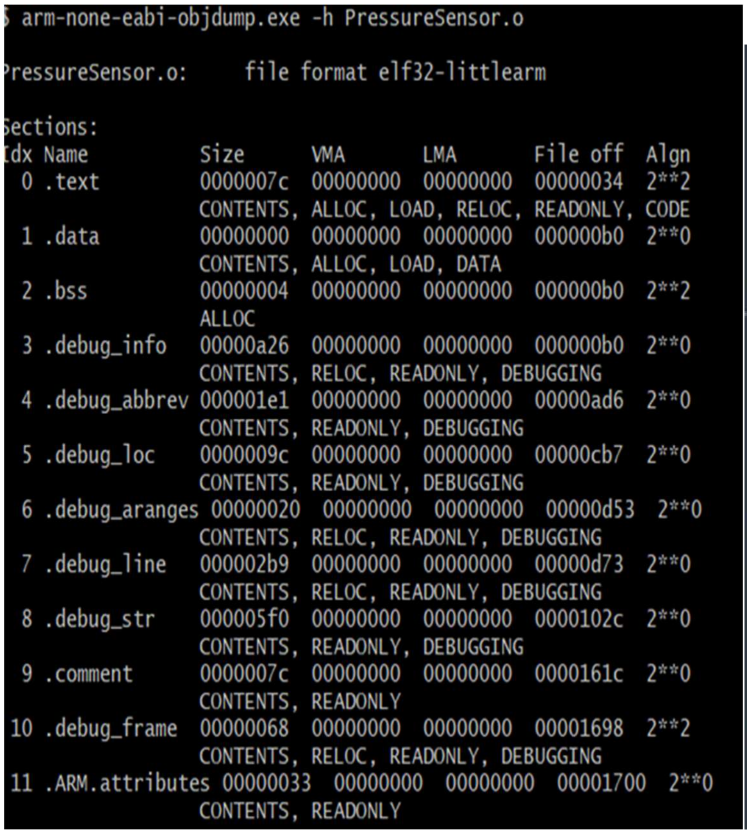


```
$ arm-none-eabi-nm main.o
00000001 C AlaramActuator_ID
         U AlaramActuator_Init
         U ALARM_ACT_STATE
00000001 C Alarm_Monitor_ID
         U ALARM_MONITOR_STATE
         U GPIO_INITIALIZATION
00000030 T main
         U MAIN_ALGO_STATE
00000001 C Main_Algorithm_ID
00000001 C PressureSensor_ID
         U PressureSensor_Init
         U PSENSOR_STATE
00000000 T Setup
         U ST_AlarmOFF
         U ST_HighPressure_detection
```

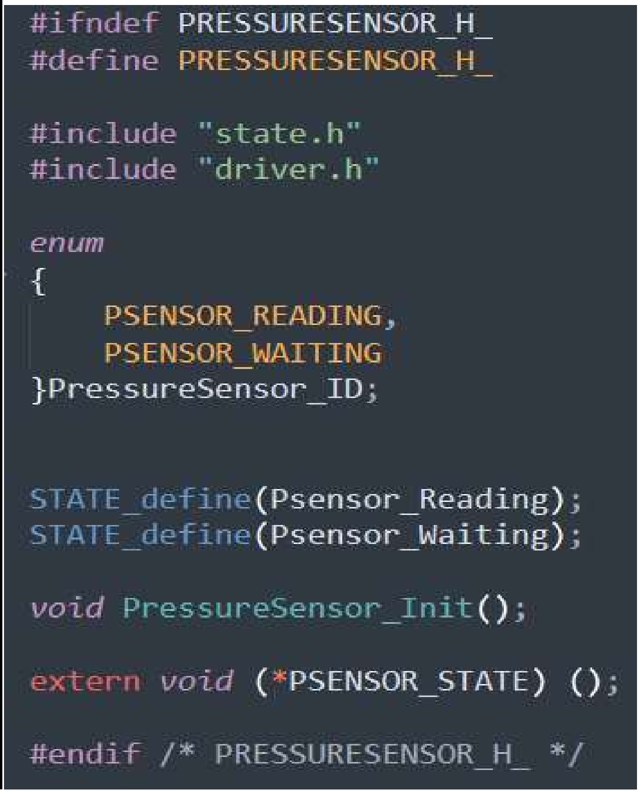
- Pressure Sensor:



```c
#include "PressureSensor.h"


int Sensor_Reading = 0;

void (*PSENSOR_STATE) ();

void PressureSensor_Init()
{
    PSENSOR_STATE = STATE(Psensor_Reading);
}

STATE_define(Psensor_Reading)
{
    PressureSensor_ID = PSENSOR_READING;

    Sensor_Reading = getPressureVal();
    Set_PressureValue(Sensor_Reading);


    PSENSOR_STATE = STATE(Psensor_Waiting);
}


STATE_define(Psensor_Waiting)
{
    PressureSensor_ID = PSENSOR_WAITING;

    Delay(10000);

    PSENSOR_STATE = STATE(Psensor_Reading);

}
```

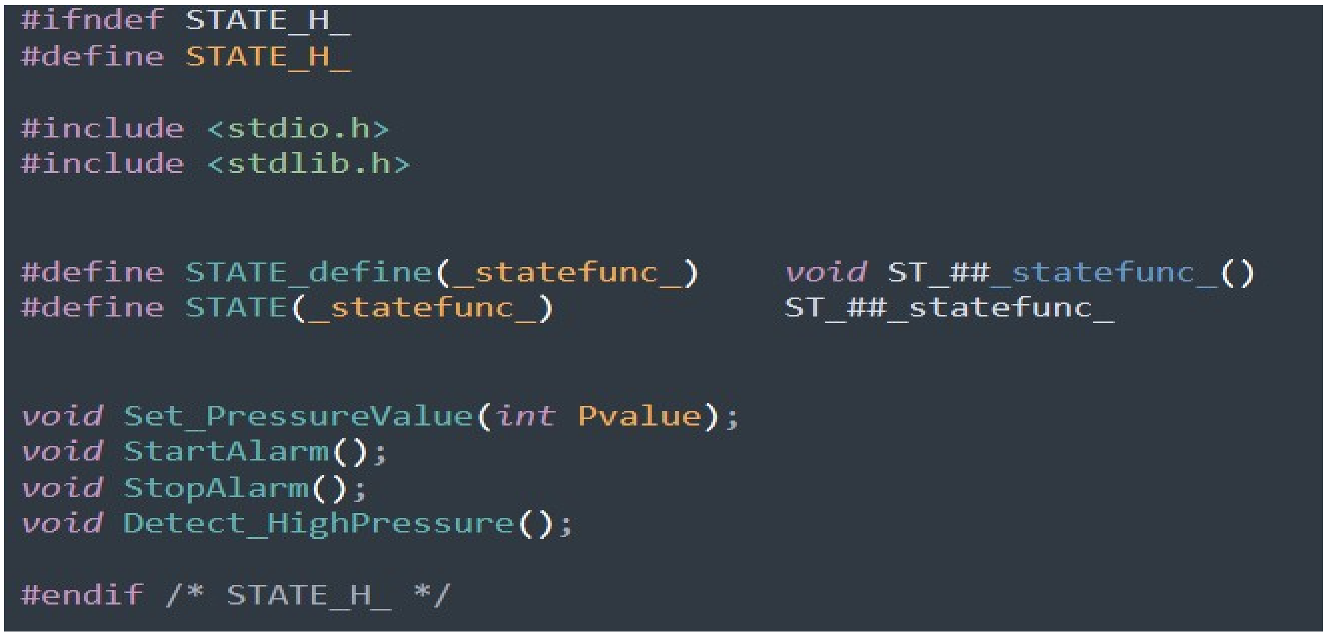


```
$ arm-none-eabi-objdump.exe -h PressureSensor.o

ressureSensor.o:     file format elf32-littlearm

ections:
dx Name          Size      VMA       LMA       File off  Algn
 0 .text         0000007c  00000000  00000000  00000034  2**2
                 CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data         00000000  00000000  00000000  000000b0  2**0
                 CONTENTS, ALLOC, LOAD, DATA
 2 .bss          00000004  00000000  00000000  000000b0  2**2
                 ALLOC
 3 .debug_info   00000a26  00000000  00000000  000000b0  2**0
                 CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001e1  00000000  00000000  00000ad6  2**0
                 CONTENTS, READONLY, DEBUGGING
 5 .debug_loc    0000009c  00000000  00000000  00000cb7  2**0
                 CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020  00000000  00000000  00000d53  2**0
                 CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   000002b9  00000000  00000000  00000d73  2**0
                 CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    000005f0  00000000  00000000  0000102c  2**0
                 CONTENTS, READONLY, DEBUGGING
 9 .comment      0000007c  00000000  00000000  0000161c  2**0
                 CONTENTS, READONLY
10 .debug_frame  00000068  00000000  00000000  00001698  2**2
                 CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033  00000000  00000000  00001700  2**0
                 CONTENTS, READONLY
```



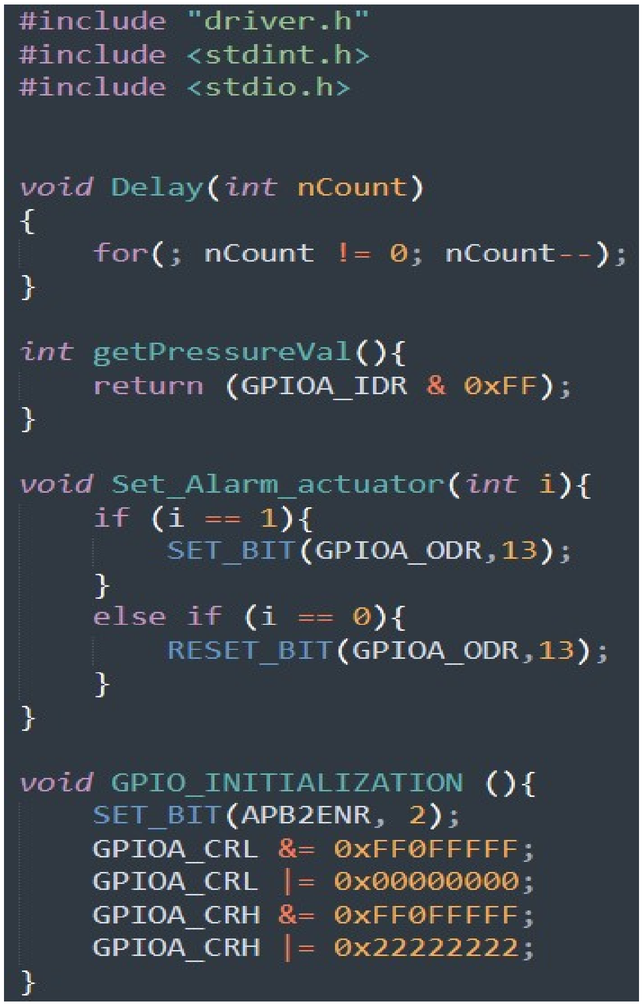
```c
#ifndef PRESSURESENSOR_H_
#define PRESSURESENSOR_H_

#include "state.h"
#include "driver.h"

enum
{
    PSENSOR_READING,
    PSENSOR_WAITING
}PressureSensor_ID;


STATE_define(Psensor_Reading);
STATE_define(Psensor_Waiting);

void PressureSensor_Init();

extern void (*PSENSOR_STATE) ();

#endif /* PRESSURESENSOR_H_ */
```

- State.h:

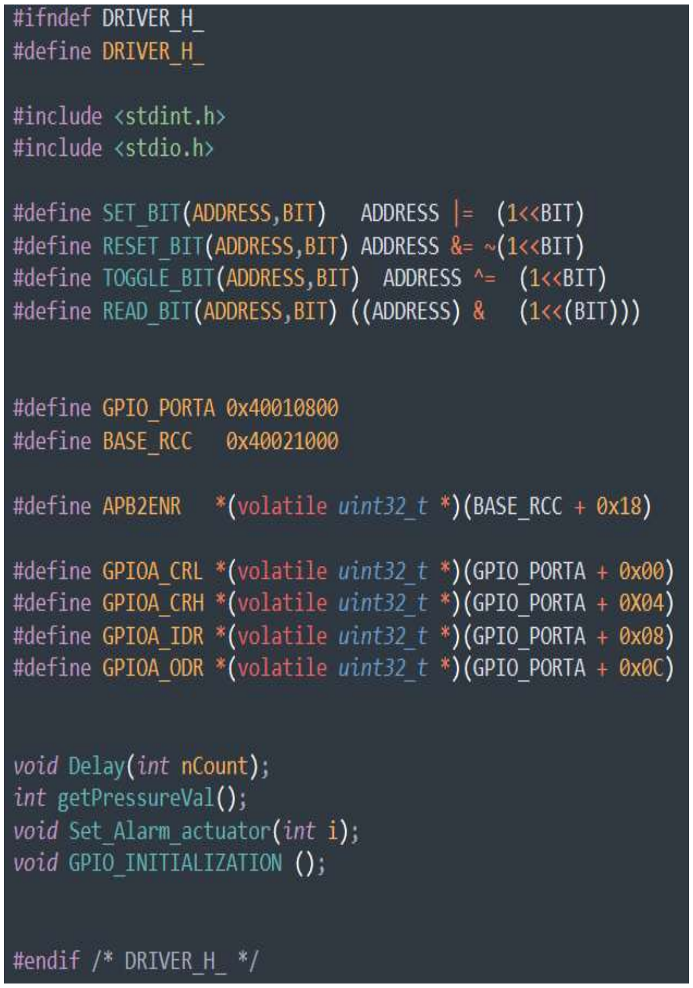


```c
#ifndef STATE_H_
#define STATE_H_

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


#define STATE_define(_statefunc_)     void ST_##_statefunc_()
#define STATE(_statefunc_)            ST_##_statefunc_


void Set_PressureValue(int Pvalue);
void StartAlarm();
void StopAlarm();
void Detect_HighPressure();

#endif /* STATE_H_ */
```

- Driver:

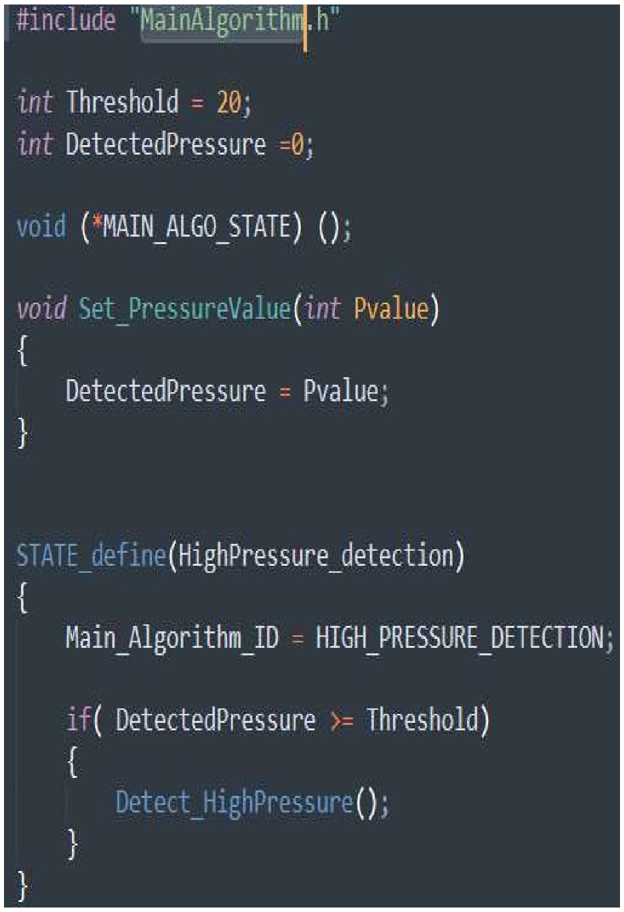


```c
#include "driver.h"
#include <stdint.h>
#include <stdio.h>


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;
}
```

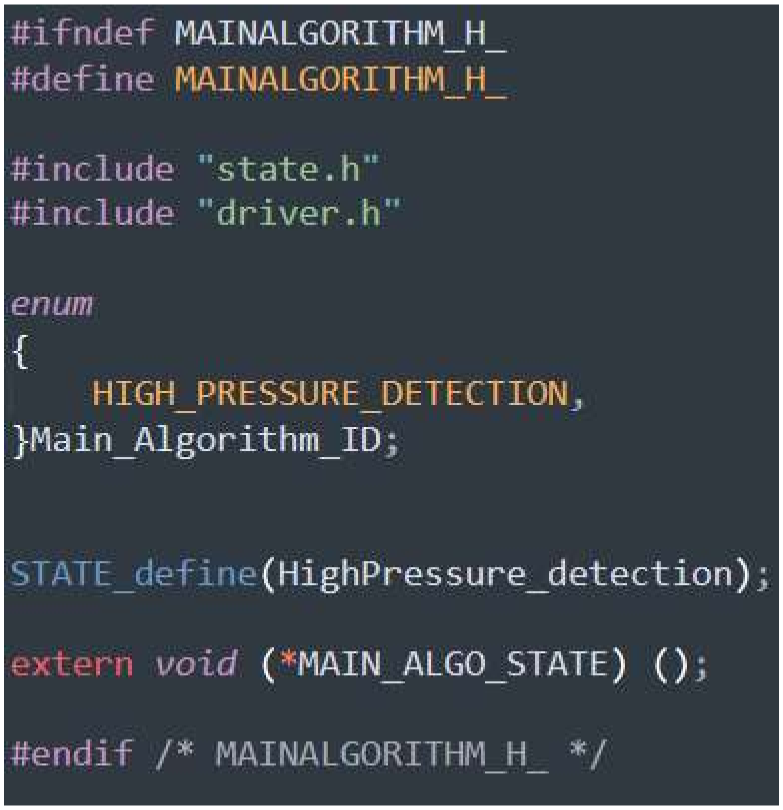


```c
#ifndef DRIVER_H_
#define DRIVER_H_

#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 ();


#endif /* DRIVER_H_ */
```

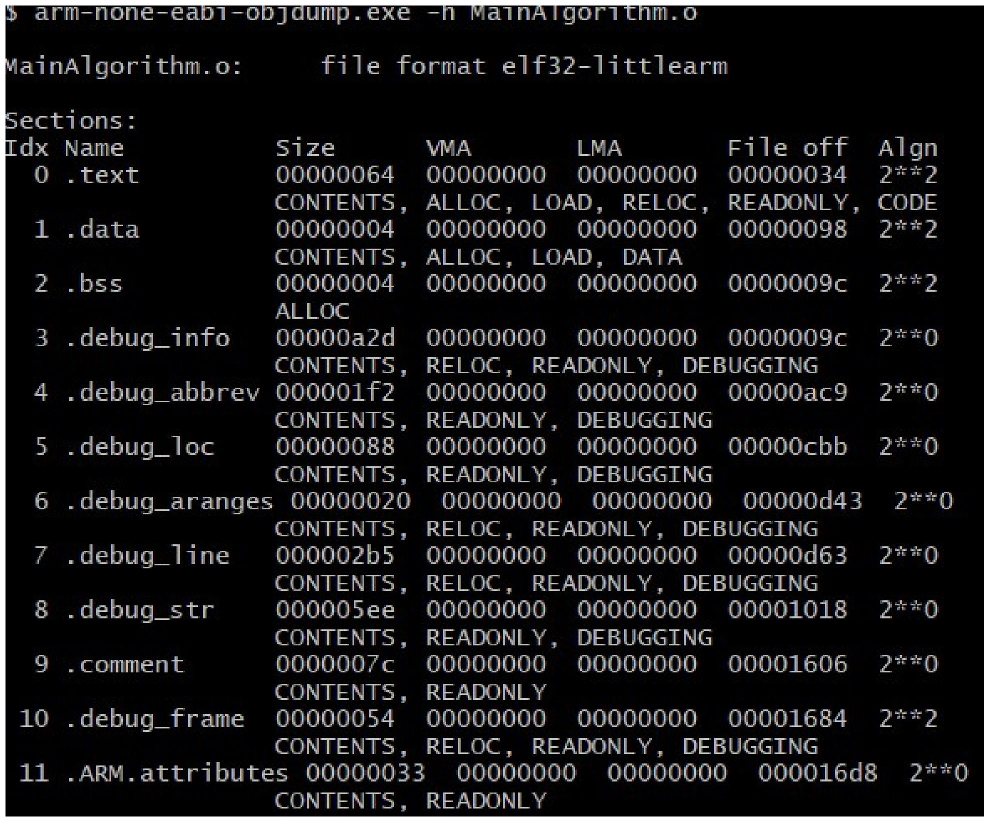
- Main Algorithm:



```c
#include "MainAlgorithm.h"

int Threshold = 20;
int DetectedPressure =0;

void (*MAIN_ALGO_STATE) ();

void Set_PressureValue(int Pvalue)
{
    DetectedPressure = Pvalue;
}


STATE_define(HighPressure_detection)
{
    Main_Algorithm_ID = HIGH_PRESSURE_DETECTION;

    if( DetectedPressure >= Threshold)
    {
        Detect_HighPressure();
    }
}
```

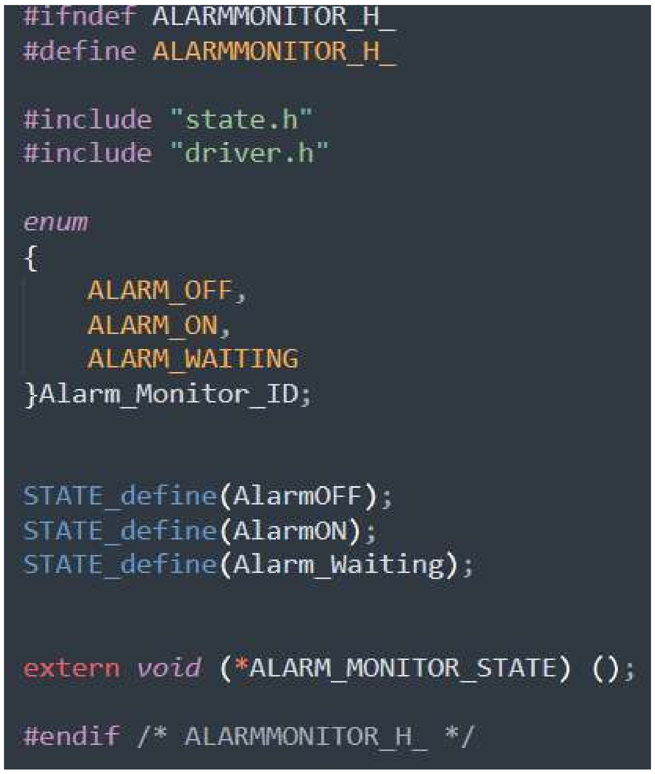


```c
#ifndef MAINALGORITHM_H_
#define MAINALGORITHM_H_

#include "state.h"
#include "driver.h"

enum
{
    HIGH_PRESSURE_DETECTION,
}Main_Algorithm_ID;


STATE_define(HighPressure_detection);

extern void (*MAIN_ALGO_STATE) ();

#endif /* MAINALGORITHM_H_ */
```

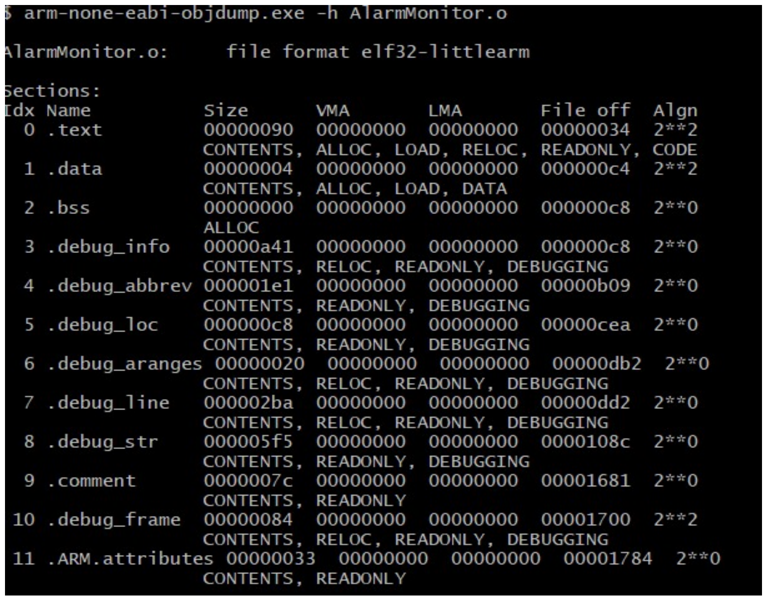


```
$ arm-none-eabi-objdump.exe -h MainAlgorithm.o

MainAlgorithm.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000064  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000004  00000000  00000000  00000098  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  0000009c  2**2
                  ALLOC
  3 .debug_info   00000a2d  00000000  00000000  0000009c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001f2  00000000  00000000  00000ac9  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000088  00000000  00000000  00000cbb  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000d43  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002b5  00000000  00000000  00000d63  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000005ee  00000000  00000000  00001018  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  00001606  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000054  00000000  00000000  00001684  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  000016d8  2**0
                  CONTENTS, READONLY
```

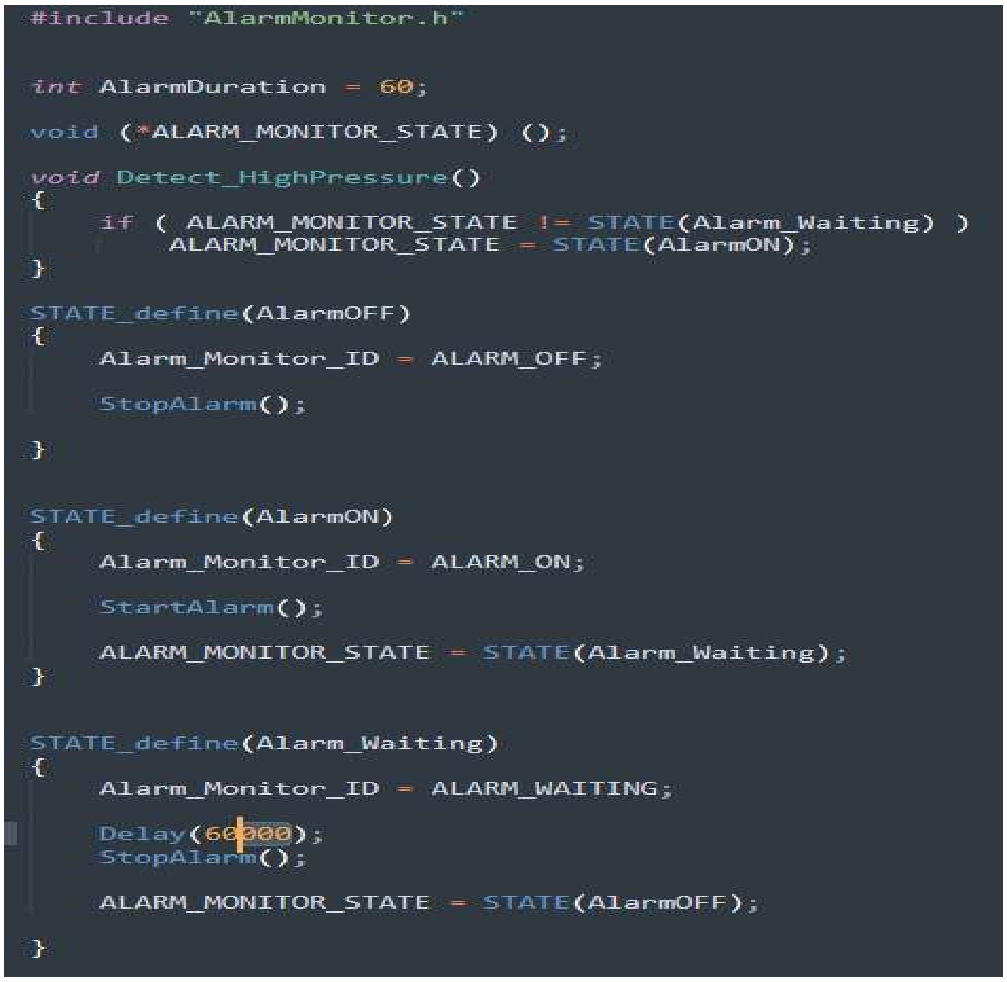
- Alarm Monitor:



```c
#ifndef ALARMMONITOR_H_
#define ALARMMONITOR_H_

#include "state.h"
#include "driver.h"

enum
{
    ALARM_OFF,
    ALARM_ON,
    ALARM_WAITING
}Alarm_Monitor_ID;


STATE_define(AlarmOFF);
STATE_define(AlarmON);
STATE_define(Alarm_Waiting);


extern void (*ALARM_MONITOR_STATE) ();

#endif /* ALARMMONITOR_H_ */
```

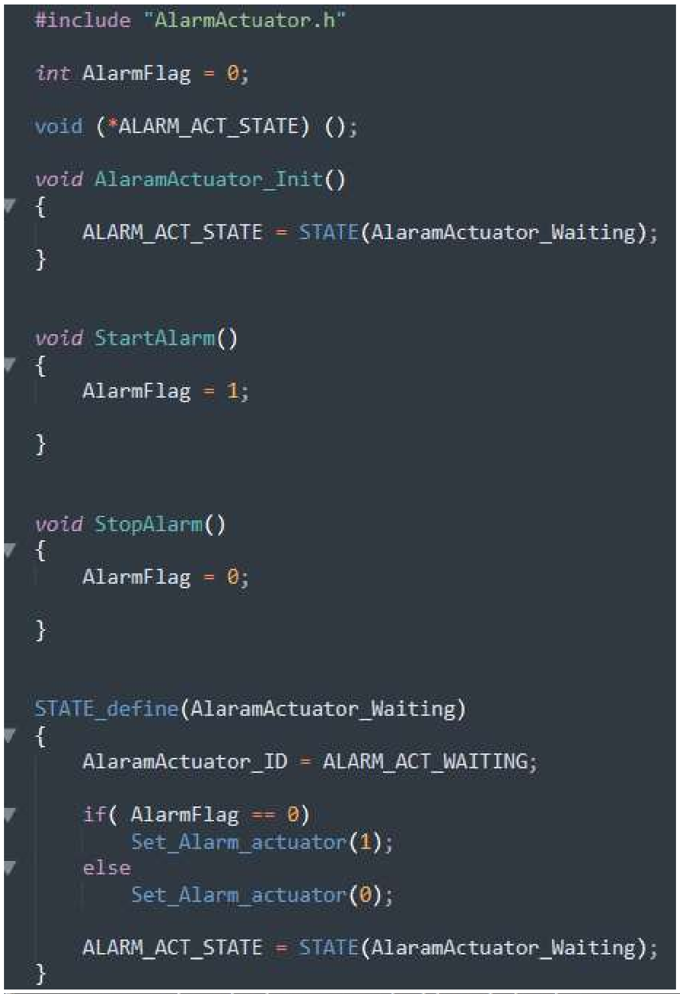


```
$ arm-none-eabi-objdump.exe -h AlarmMonitor.o

AlarmMonitor.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         00000004  00000000  00000000  000000c4  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000c8  2**0
                  ALLOC
  3 .debug_info   00000a41  00000000  00000000  000000c8  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001e1  00000000  00000000  00000b09  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000000c8  00000000  00000000  00000cea  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000db2  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002ba  00000000  00000000  00000dd2  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000005f5  00000000  00000000  0000108c  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  00001681  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000084  00000000  00000000  00001700  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001784  2**0
                  CONTENTS, READONLY
```

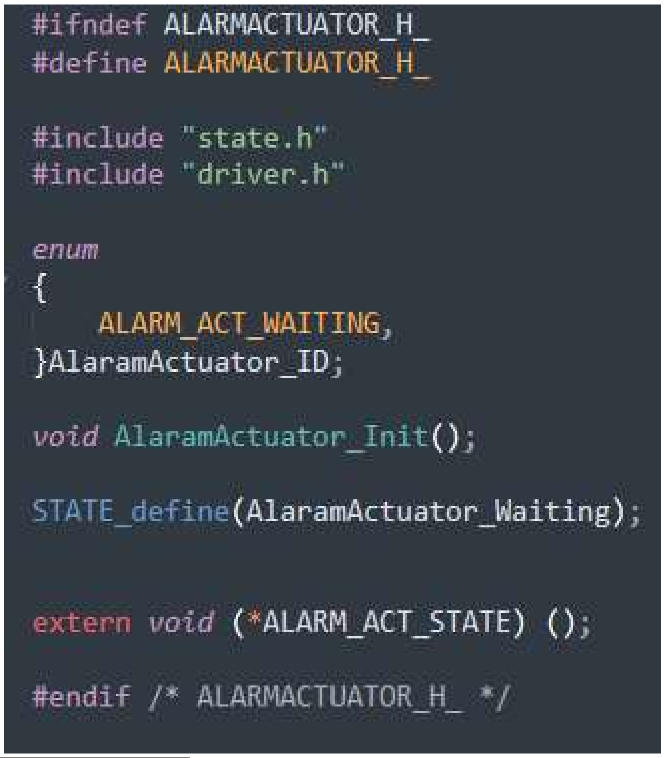


```c
#include "AlarmMonitor.h"


int AlarmDuration = 60;

void (*ALARM_MONITOR_STATE) ();

void Detect_HighPressure()
{
    if ( ALARM_MONITOR_STATE != STATE(Alarm_Waiting) )
        ALARM_MONITOR_STATE = STATE(AlarmON);
}

STATE_define(AlarmOFF)
{
    Alarm_Monitor_ID = ALARM_OFF;

    StopAlarm();

}

STATE_define(AlarmON)
{
    Alarm_Monitor_ID = ALARM_ON;

    StartAlarm();

    ALARM_MONITOR_STATE = STATE(Alarm_Waiting);
}

STATE_define(Alarm_Waiting)
{
    Alarm_Monitor_ID = ALARM_WAITING;

    Delay(60000);
    StopAlarm();

    ALARM_MONITOR_STATE = STATE(AlarmOFF);

}
```

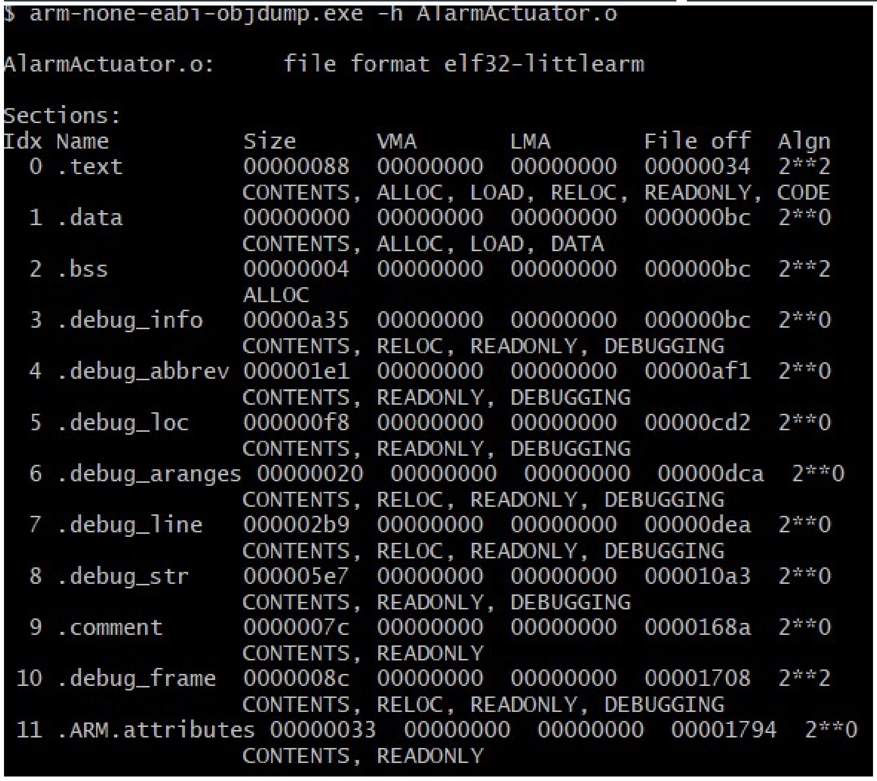
- Alarm Actuator:



```
#include "AlarmActuator.h"

int AlarmFlag = 0;

void (*ALARM_ACT_STATE) ();

void AlaramActuator_Init()
{
    ALARM_ACT_STATE = STATE(AlaramActuator_Waiting);
}


void StartAlarm()
{
    AlarmFlag = 1;

}


void StopAlarm()
{
    AlarmFlag = 0;

}


STATE_define(AlaramActuator_Waiting)
{
    AlaramActuator_ID = ALARM_ACT_WAITING;

    if( AlarmFlag == 0)
        Set_Alarm_actuator(1);
    else
        Set_Alarm_actuator(0);

    ALARM_ACT_STATE = STATE(AlaramActuator_Waiting);
}
```

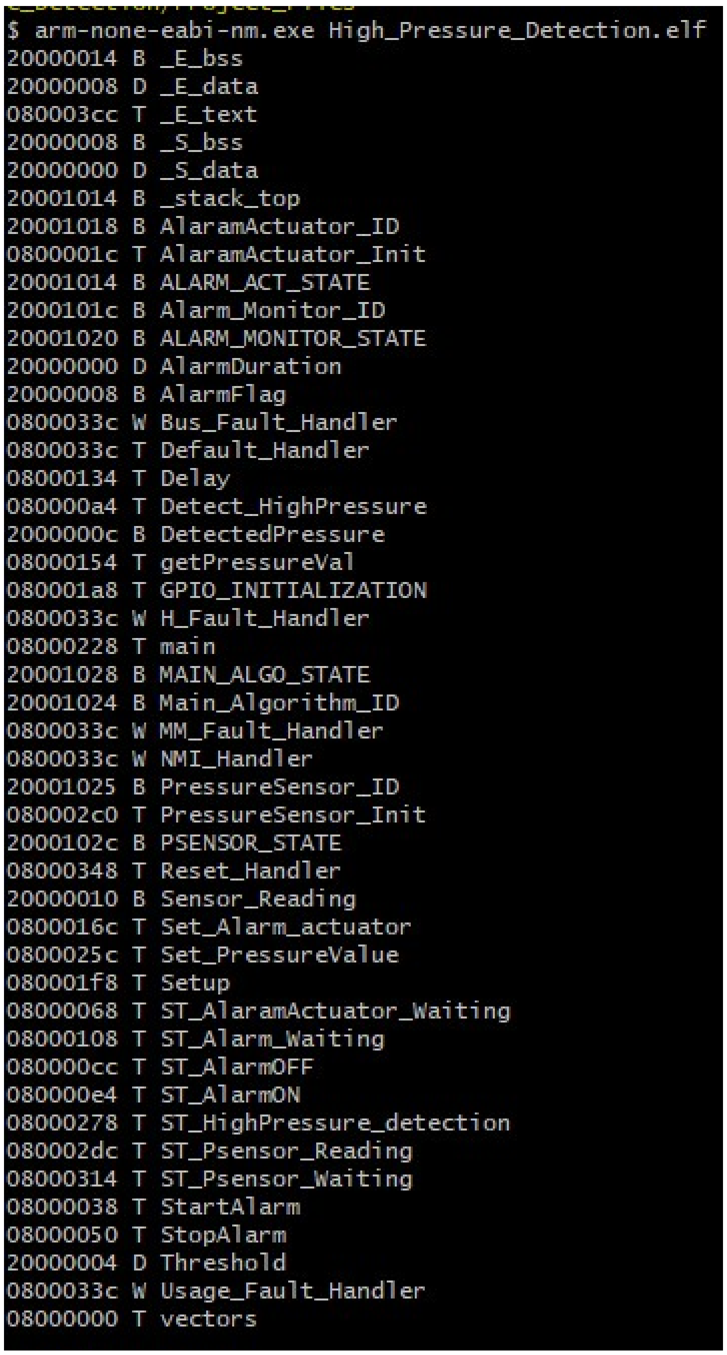


```
#ifndef ALARMACTUATOR_H_
#define ALARMACTUATOR_H_

#include "state.h"
#include "driver.h"

enum
{
    ALARM_ACT_WAITING,
}AlaramActuator_ID;

void AlaramActuator_Init();

STATE_define(AlaramActuator_Waiting);


extern void (*ALARM_ACT_STATE) ();

#endif /* ALARMACTUATOR_H_ */
```



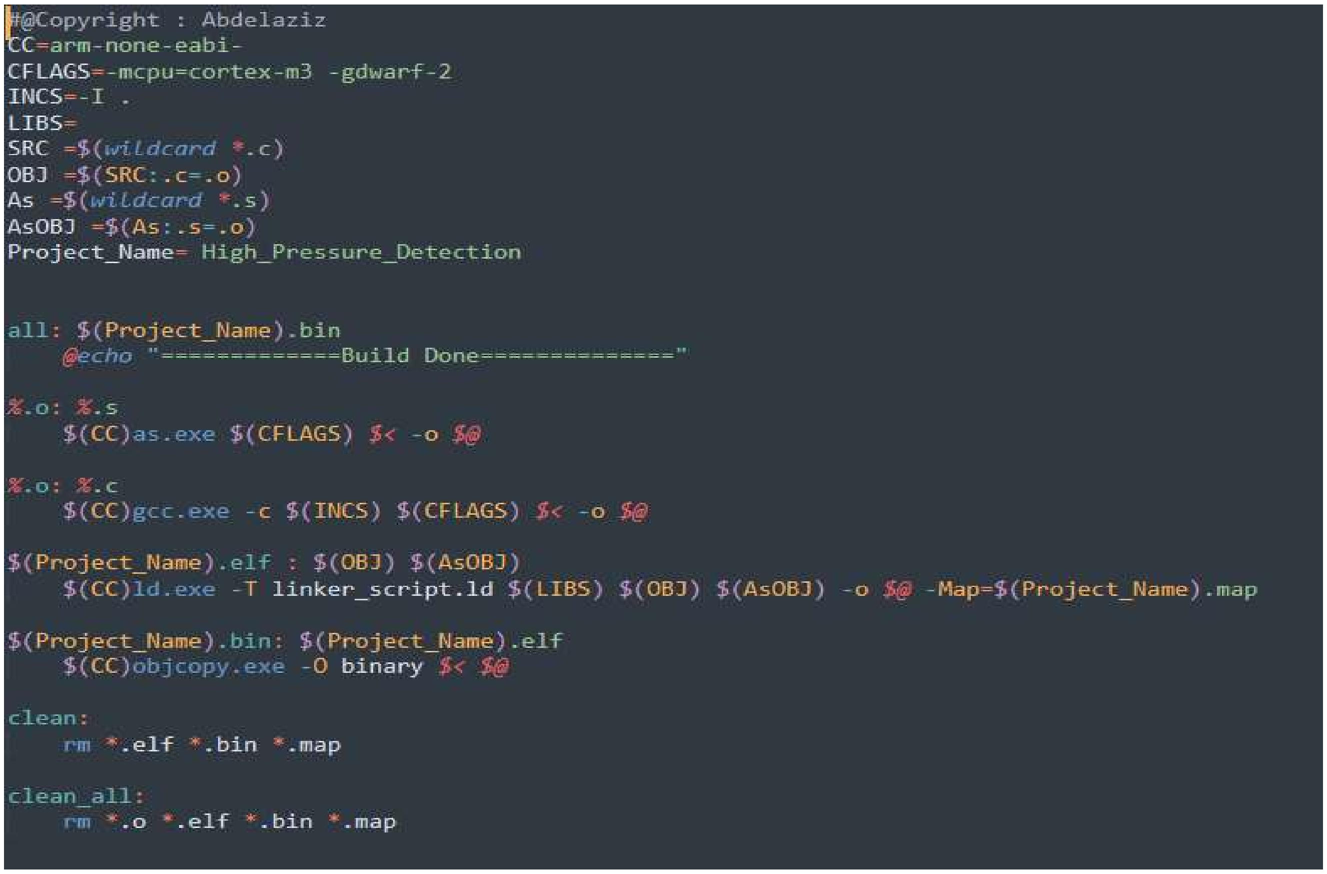
```
$ arm-none-eabi-objdump.exe -h AlarmActuator.o

AlarmActuator.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000088  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000bc  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  000000bc  2**2
                  ALLOC
  3 .debug_info   00000a35  00000000  00000000  000000bc  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001e1  00000000  00000000  00000af1  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000000f8  00000000  00000000  00000cd2  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000dca  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002b9  00000000  00000000  00000dea  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000005e7  00000000  00000000  000010a3  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  0000168a  2**0
                  CONTENTS, READONLY
 10 .debug_frame  0000008c  00000000  00000000  00001708  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001794  2**0
                  CONTENTS, READONLY
```

- ELF Section Table:



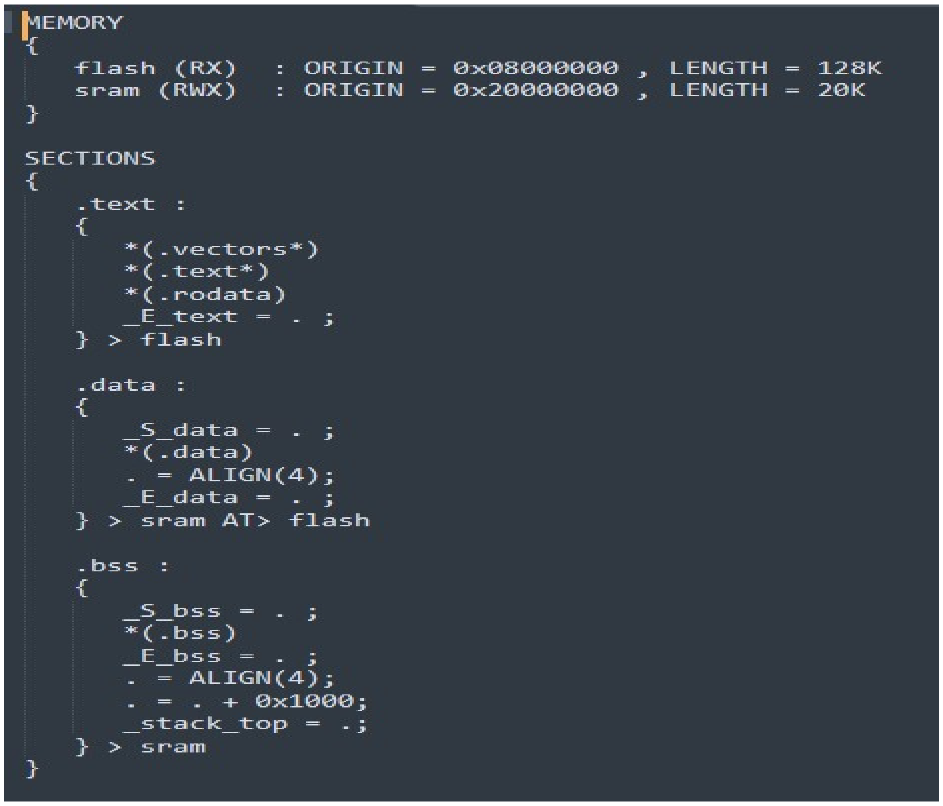
```
$ arm-none-eabi-nm.exe High_Pressure_Detection.elf
20000014 B _E_bss
20000008 D _E_data
080003cc T _E_text
20000008 B _S_bss
20000000 D _S_data
20001014 B _stack_top
20001018 B AlaramActuator_ID
0800001c T AlaramActuator_Init
20001014 B ALARM_ACT_STATE
2000101c B Alarm_Monitor_ID
20001020 B ALARM_MONITOR_STATE
20000000 D AlarmDuration
20000008 B AlarmFlag
0800033c W Bus_Fault_Handler
0800033c T Default_Handler
08000134 T Delay
080000a4 T Detect_HighPressure
2000000c B DetectedPressure
08000154 T getPressureVal
080001a8 T GPIO_INITIALIZATION
0800033c W H_Fault_Handler
08000228 T main
20001028 B MAIN_ALGO_STATE
20001024 B Main_Algorithm_ID
0800033c W MM_Fault_Handler
0800033c W NMI_Handler
20001025 B PressureSensor_ID
080002c0 T PressureSensor_Init
2000102c B PSENSOR_STATE
08000348 T Reset_Handler
20000010 B Sensor_Reading
0800016c T Set_Alarm_actuator
0800025c T Set_PressureValue
080001f8 T Setup
08000068 T ST_AlaramActuator_Waiting
08000108 T ST_Alarm_Waiting
080000cc T ST_AlarmOFF
080000e4 T ST_AlarmON
08000278 T ST_HighPressure_detection
080002dc T ST_Psensor_Reading
08000314 T ST_Psensor_Waiting
08000038 T StartAlarm
08000050 T StopAlarm
20000004 D Threshold
0800033c W Usage_Fault_Handler
08000000 T vectors
```

- MakeFile:



```
#@Copyright : Abdelaziz
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)
Project_Name= High_Pressure_Detection


all: $(Project_Name).bin
    @echo "============Build 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 $@ -Map=$(Project_Name).map

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

clean:
    rm *.elf *.bin *.map

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

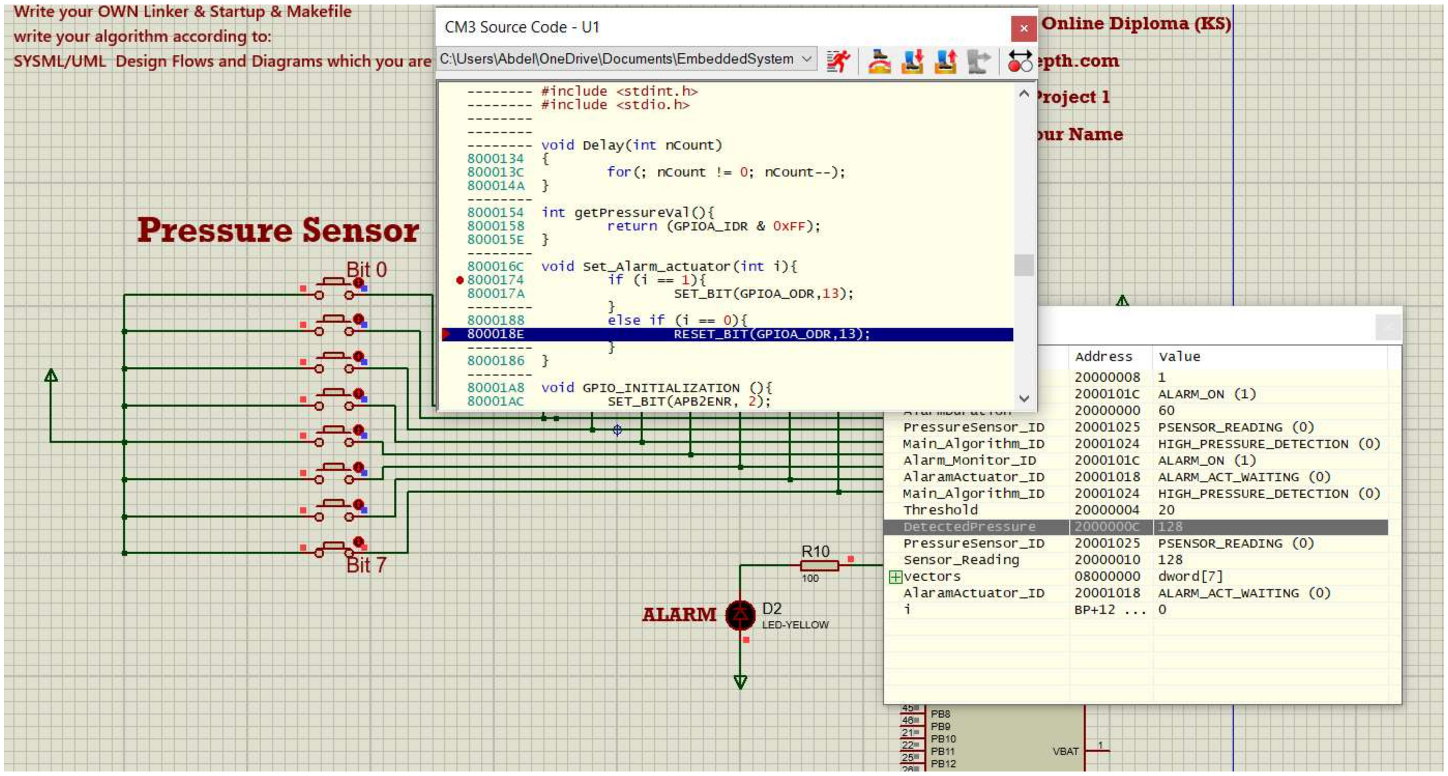
- LinkerScript:



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

Proteus Simulation:

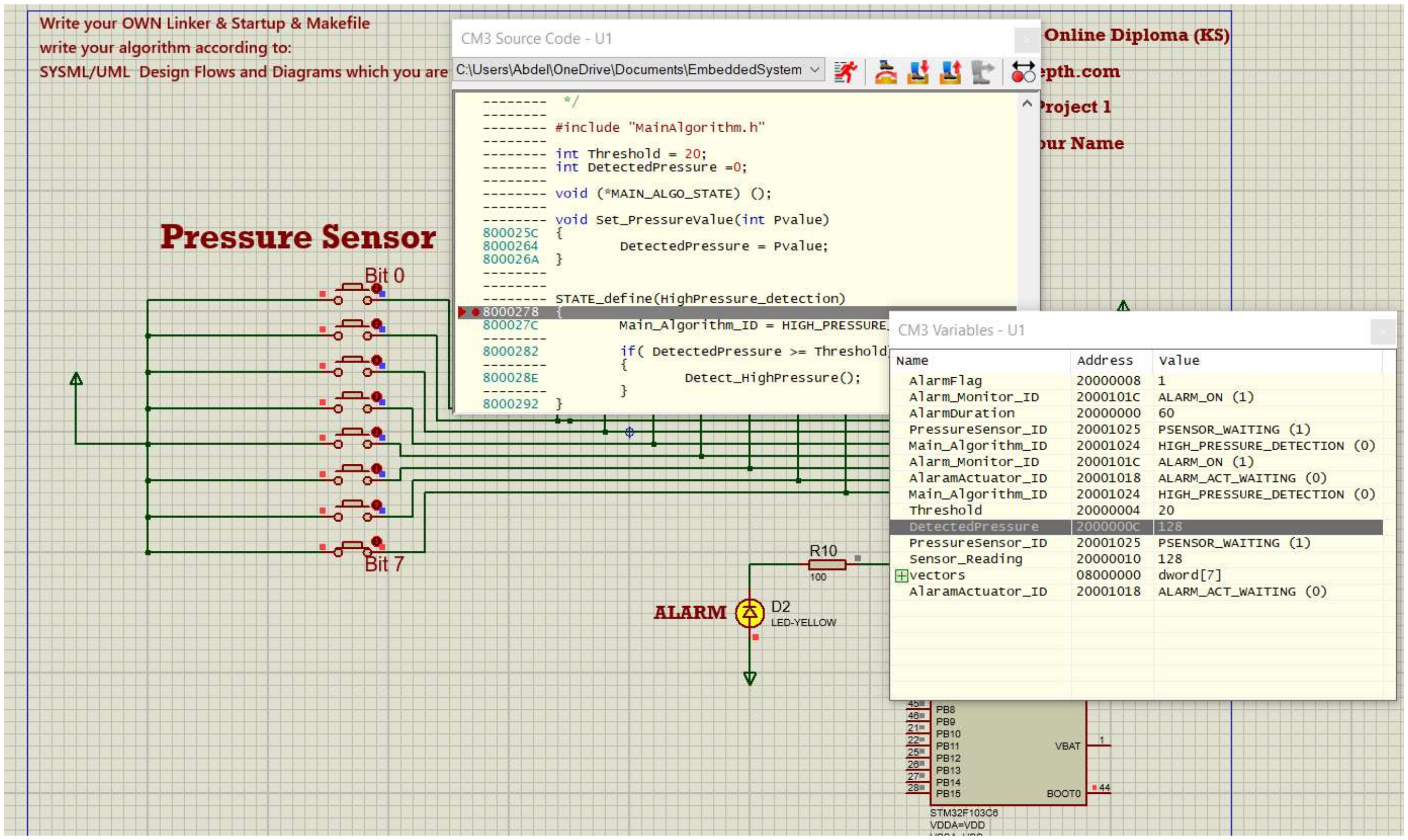
- Alarm ON:



Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are
Online Diploma (KS)
epth.com
Project 1
our Name
Pressure Sensor
Bit 0
Bit 7
R10
100
ALARM
D2
LED-YELLOW
PB8
PB9
PB10
PB11
PB12
VBAT
CM3 Source Code - U1
C:\Users\Abdel\OneDrive\Documents\EmbeddedSystem

```
-------- #include <stdint.h>
-------- #include <stdio.h>
--------
--------
-------- void Delay(int nCount)
8000134  {
                 for(; nCount != 0; nCount--);
800014A  }
--------
8000154  int getPressureVal(){
8000158          return (GPIOA_IDR & 0xFF);
800015E  }
--------
800016C  void Set_Alarm_actuator(int i){
8000174          if (i == 1){
800017A                  SET_BIT(GPIOA_ODR,13);
--------         }
8000188          else if (i == 0){
800018E                  RESET_BIT(GPIOA_ODR,13);
--------         }
8000186  }
--------
80001A8  void GPIO_INITIALIZATION (){
80001AC          SET_BIT(APB2ENR, 2);
```

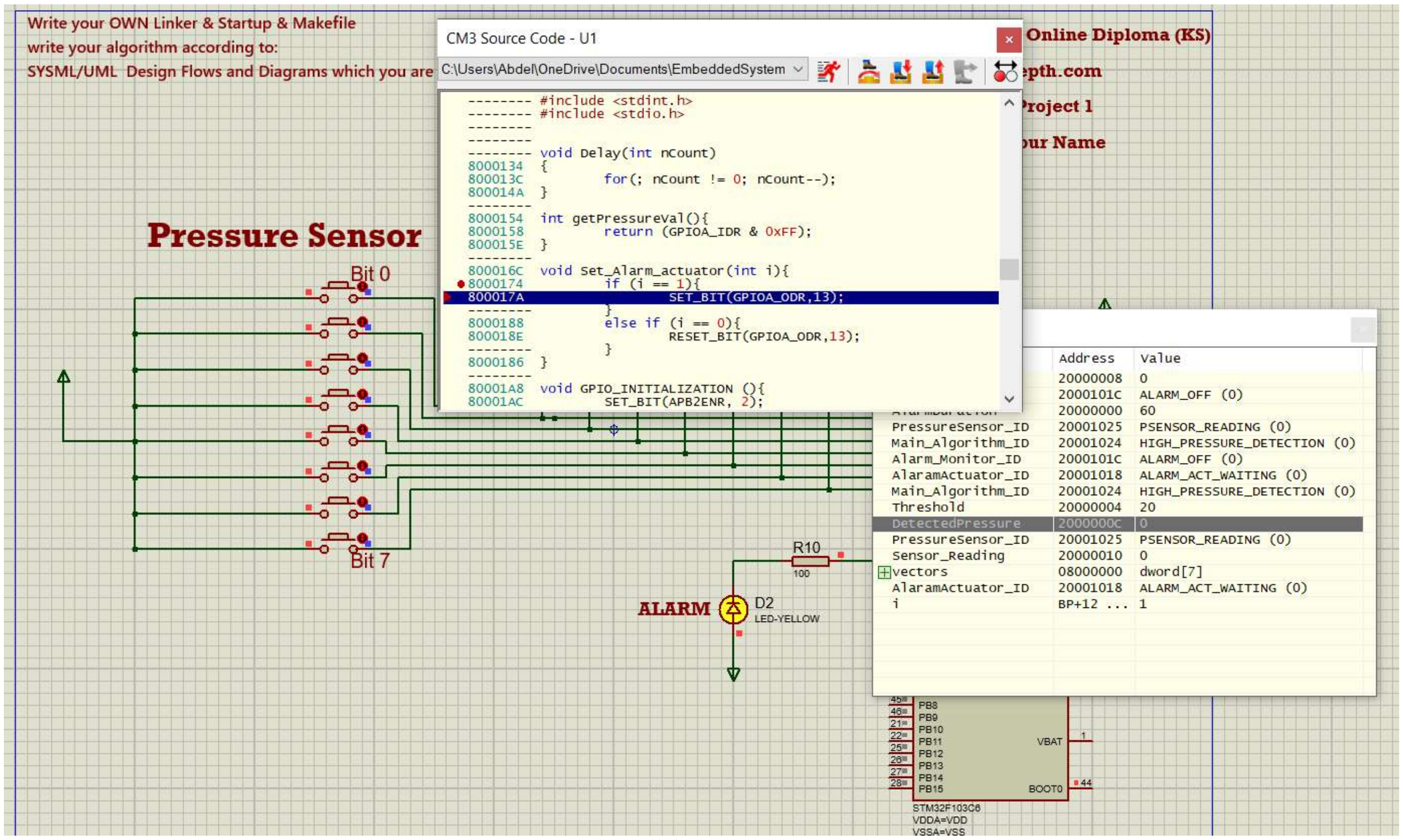
	Address	Value
	20000008	1
	2000101c	ALARM_ON (1)
	20000000	60
PressureSensor_ID	20001025	PSENSOR_READING (0)
Main_Algorithm_ID	20001024	HIGH_PRESSURE_DETECTION (0)
Alarm_Monitor_ID	2000101c	ALARM_ON (1)
AlaramActuator_ID	20001018	ALARM_ACT_WAITING (0)
Main_Algorithm_ID	20001024	HIGH_PRESSURE_DETECTION (0)
Threshold	20000004	20
DetectedPressure	2000000c	128
PressureSensor_ID	20001025	PSENSOR_READING (0)
Sensor_Reading	20000010	128
vectors	08000000	dword[7]
AlaramActuator_ID	20001018	ALARM_ACT_WAITING (0)
i	BP+12 ...	0



```
--------  */
--------
-------- #include "MainAlgorithm.h"
--------
-------- int Threshold = 20;
-------- int DetectedPressure =0;
--------
-------- void (*MAIN_ALGO_STATE) ();
--------
-------- void Set_PressureValue(int Pvalue)
800025C  {
8000264          DetectedPressure = Pvalue;
800026A  }
--------
--------
-------- STATE_define(HighPressure_detection)
8000278  {
800027C          Main_Algorithm_ID = HIGH_PRESSURE_
--------
8000282          if( DetectedPressure >= Threshold
--------         {
800028E                  Detect_HighPressure();
--------         }
8000292  }
```

Name	Address	Value
AlarmFlag	20000008	1
Alarm_Monitor_ID	2000101c	ALARM_ON (1)
AlarmDuration	20000000	60
PressureSensor_ID	20001025	PSENSOR_WAITING (1)
Main_Algorithm_ID	20001024	HIGH_PRESSURE_DETECTION (0)
Alarm_Monitor_ID	2000101c	ALARM_ON (1)
AlaramActuator_ID	20001018	ALARM_ACT_WAITING (0)
Main_Algorithm_ID	20001024	HIGH_PRESSURE_DETECTION (0)
Threshold	20000004	20
DetectedPressure	2000000c	128
PressureSensor_ID	20001025	PSENSOR_WAITING (1)
Sensor_Reading	20000010	128
vectors	08000000	dword[7]
AlaramActuator_ID	20001018	ALARM_ACT_WAITING (0)

- Alarm OFF:



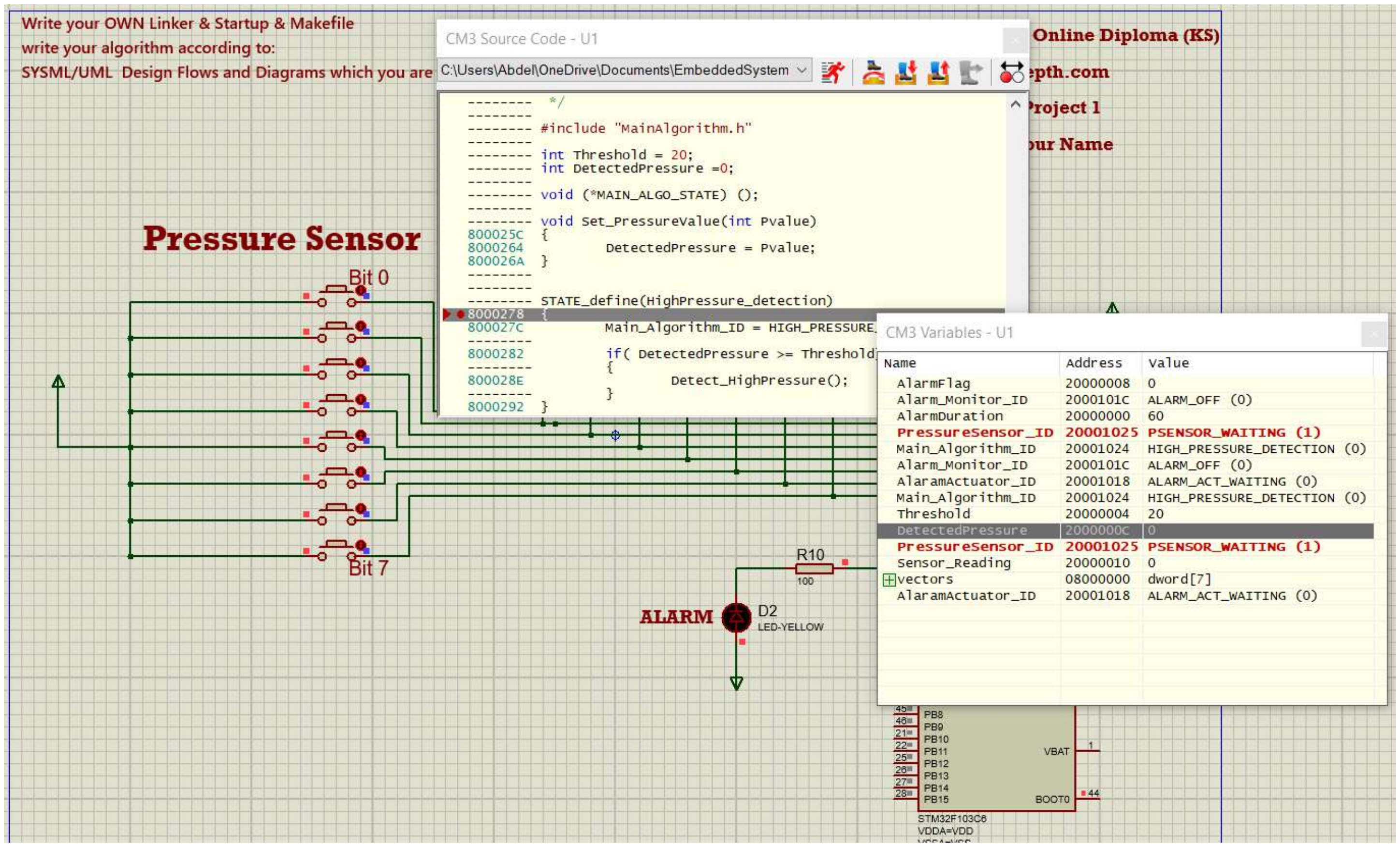
Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are
Online Diploma (KS)
epth.com
Project 1
ur Name
Pressure Sensor
Bit 0
Bit 7
R10
100
ALARM
D2
LED-YELLOW
STM32F103C6
VDDA=VDD
VSSA=VSS

CM3 Source Code - U1

C:\Users\Abdel\OneDrive\Documents\EmbeddedSystem

```
-------- #include <stdint.h>
-------- #include <stdio.h>
--------
--------
-------- void Delay(int nCount)
8000134  {
800013C          for(; nCount != 0; nCount--);
800014A  }
--------
8000154  int getPressureVal(){
8000158          return (GPIOA_IDR & 0xFF);
800015E  }
--------
800016C  void Set_Alarm_actuator(int i){
8000174          if (i == 1){
800017A                  SET_BIT(GPIOA_ODR,13);
--------         }
8000188          else if (i == 0){
800018E                  RESET_BIT(GPIOA_ODR,13);
--------         }
8000186  }
--------
80001A8  void GPIO_INITIALIZATION (){
80001AC          SET_BIT(APB2ENR, 2);
```

	Address	Value
	20000008	0
	2000101C	ALARM_OFF (0)
AlarmDuration	20000000	60
PressureSensor_ID	20001025	PSENSOR_READING (0)
Main_Algorithm_ID	20001024	HIGH_PRESSURE_DETECTION (0)
Alarm_Monitor_ID	2000101C	ALARM_OFF (0)
AlaramActuator_ID	20001018	ALARM_ACT_WAITING (0)
Main_Algorithm_ID	20001024	HIGH_PRESSURE_DETECTION (0)
Threshold	20000004	20
DetectedPressure	2000000C	0
PressureSensor_ID	20001025	PSENSOR_READING (0)
Sensor_Reading	20000010	0
⊞vectors	08000000	dword[7]
AlaramActuator_ID	20001018	ALARM_ACT_WAITING (0)
i	BP+12 ...	1



Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are
Online Diploma (KS)
epth.com
Project 1
ur Name
Pressure Sensor
Bit 0
Bit 7
R10
100
ALARM
D2
LED-YELLOW
STM32F103C6
VDDA=VDD

CM3 Source Code - U1

C:\Users\Abdel\OneDrive\Documents\EmbeddedSystem

```
--------  */
--------
-------- #include "MainAlgorithm.h"
--------
-------- int Threshold = 20;
-------- int DetectedPressure =0;
--------
-------- void (*MAIN_ALGO_STATE) ();
--------
-------- void Set_PressureValue(int Pvalue)
800025C  {
8000264          DetectedPressure = Pvalue;
800026A  }
--------
-------- STATE_define(HighPressure_detection)
8000278  {
800027C          Main_Algorithm_ID = HIGH_PRESSURE
--------
8000282          if( DetectedPressure >= Threshold
--------         {
800028E                  Detect_HighPressure();
--------         }
8000292  }
```

CM3 Variables - U1

Name	Address	Value
AlarmFlag	20000008	0
Alarm_Monitor_ID	2000101C	ALARM_OFF (0)
AlarmDuration	20000000	60
PressureSensor_ID	20001025	PSENSOR_WAITING (1)
Main_Algorithm_ID	20001024	HIGH_PRESSURE_DETECTION (0)
Alarm_Monitor_ID	2000101C	ALARM_OFF (0)
AlaramActuator_ID	20001018	ALARM_ACT_WAITING (0)
Main_Algorithm_ID	20001024	HIGH_PRESSURE_DETECTION (0)
Threshold	20000004	20
DetectedPressure	2000000C	0
PressureSensor_ID	20001025	PSENSOR_WAITING (1)
Sensor_Reading	20000010	0
⊞vectors	08000000	dword[7]
AlaramActuator_ID	20001018	ALARM_ACT_WAITING (0)