Embedded System online diploma

learn-in-depth

Be Professional In Embedded System

Eng. Keroles Shenouda

www.learn-in-depth.com

PRESSURE CONTROLLER SYSTEM

First Term (Final Project 1)

Name	**Magdy Adel Isaac**
My Profile	**learn-in-depth.com/online-diploma/magdyadel608@gmail.com**
ProjectLink	

Contents:

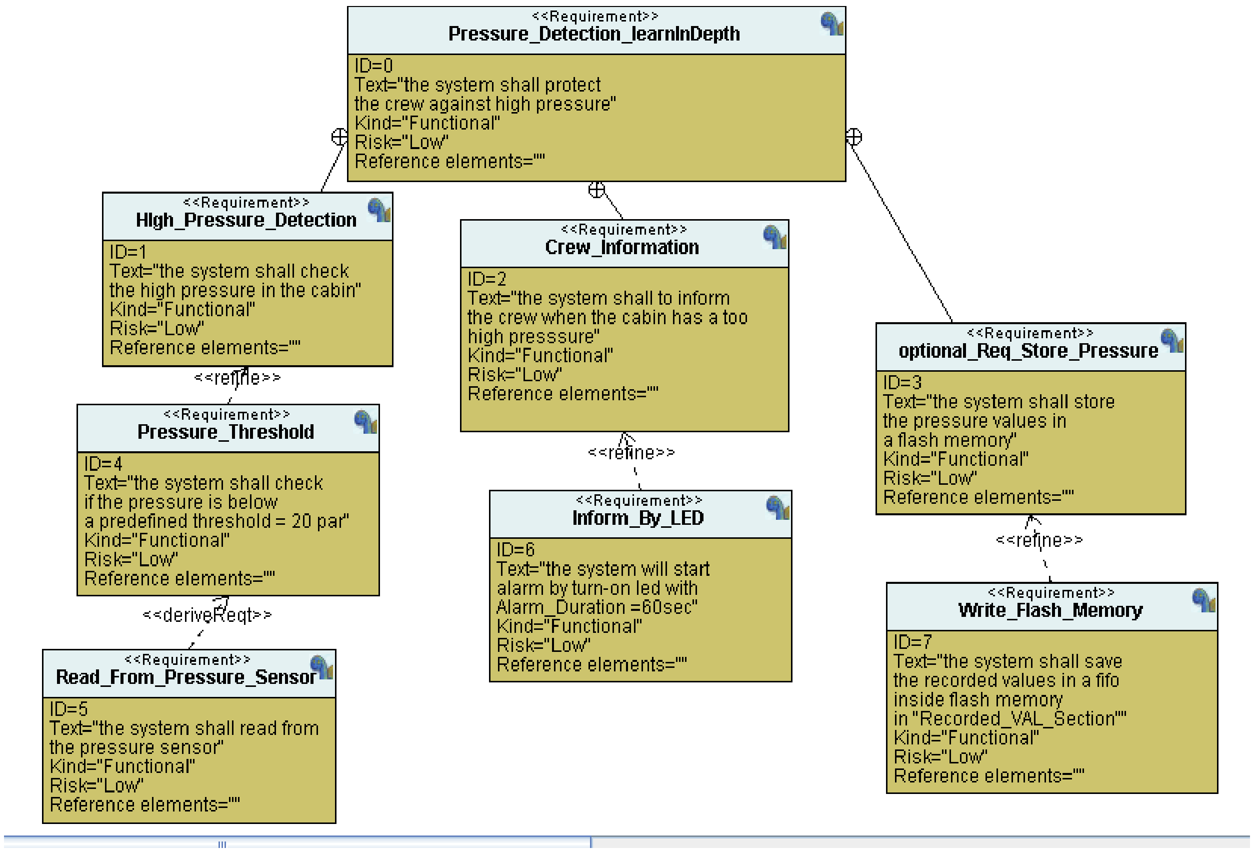
- Case Study:

- 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.

- 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.

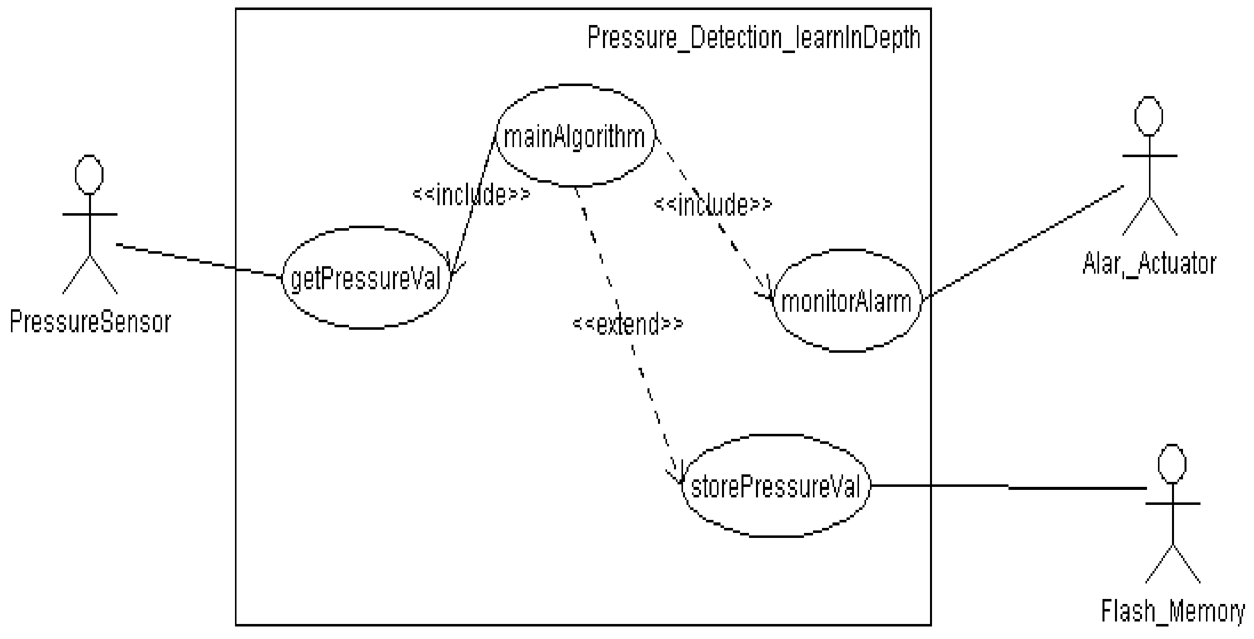
- Requirement Diagram:


<<Requirement>>
Pressure_Detection_learnInDepth
ID=0
Text="the system shall protect the crew against 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>>
Crew_Information
ID=2
Text="the system shall to inform the crew when the cabin has a too high presssure"
Kind="Functional"
Risk="Low"
Reference elements=""
<<Requirement>>
optional_Req_Store_Pressure
ID=3
Text="the system shall store the pressure values in a flash memory"
Kind="Functional"
Risk="Low"
Reference elements=""
<<refine>>
<<Requirement>>
Pressure_Threshold
ID=4
Text="the system shall check if the pressure is below a predefined threshold = 20 par"
Kind="Functional"
Risk="Low"
Reference elements=""
<<refine>>
<<Requirement>>
Inform_By_LED
ID=6
Text="the system will start alarm by turn-on led with Alarm_Duration =60sec"
Kind="Functional"
Risk="Low"
Reference elements=""
<<refine>>
<<Requirement>>
Write_Flash_Memory
ID=7
Text="the system shall save the recorded values in a fifo inside flash memory in "Recorded_VAL_Section""
Kind="Functional"
Risk="Low"
Reference elements=""
<<deriveReqt>>
<<Requirement>>
Read_From_Pressure_Sensor
ID=5
Text="the system shall read from the pressure sensor"
Kind="Functional"
Risk="Low"
Reference elements=""

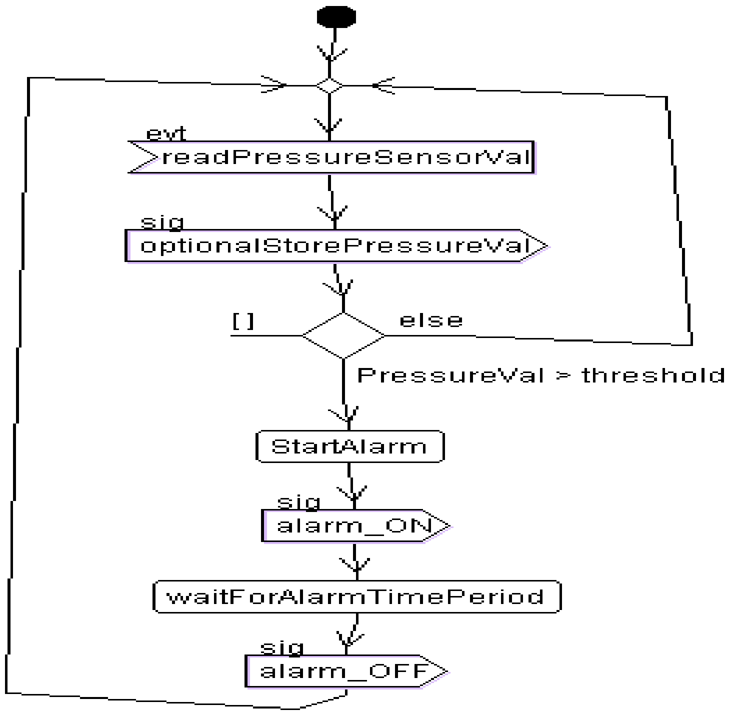
- Space Exploration (HW/SW Partitioning):

For the hardware, we have STM32 microcontroller with a cortex-m3 processor that will be more than enough for this application.

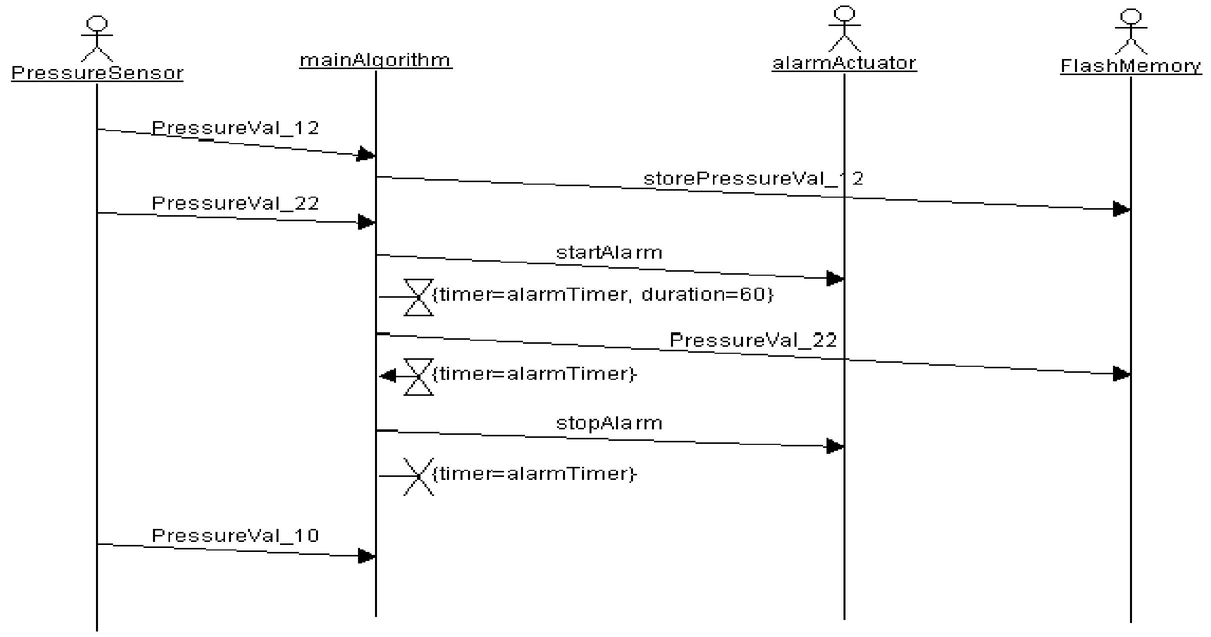
- System Analysis: Use Case Diagram: -



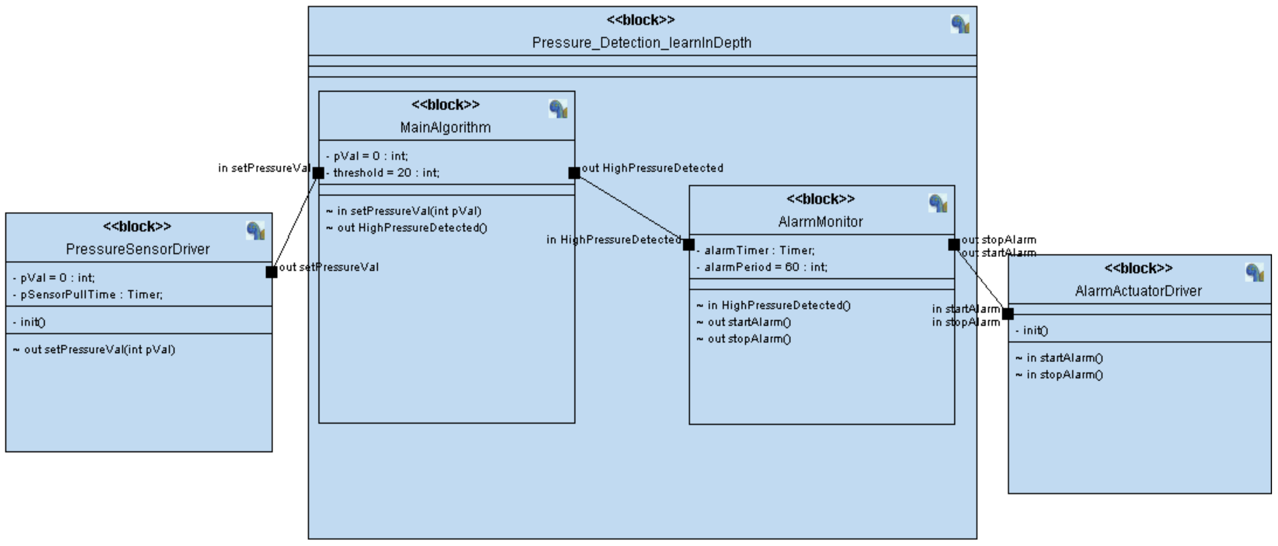
- System Analysis: Activity Diagram:



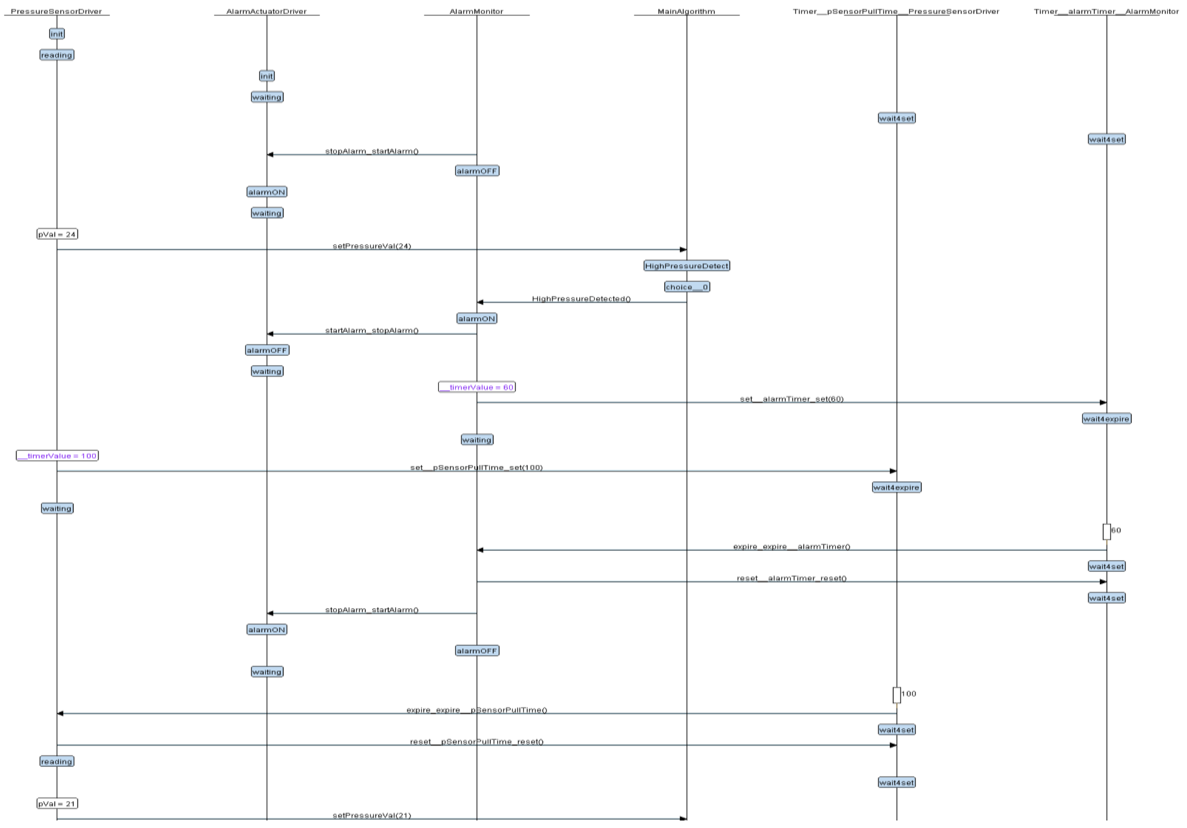
- System Analysis: Sequence Diagram:



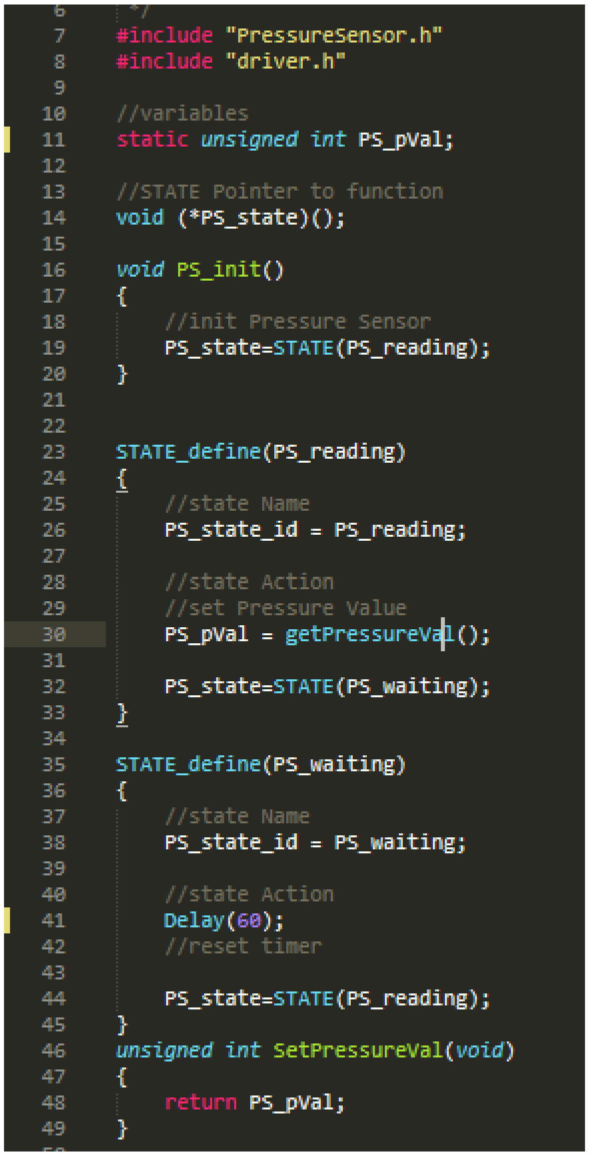
- Block Diagram:



- Final Simulation



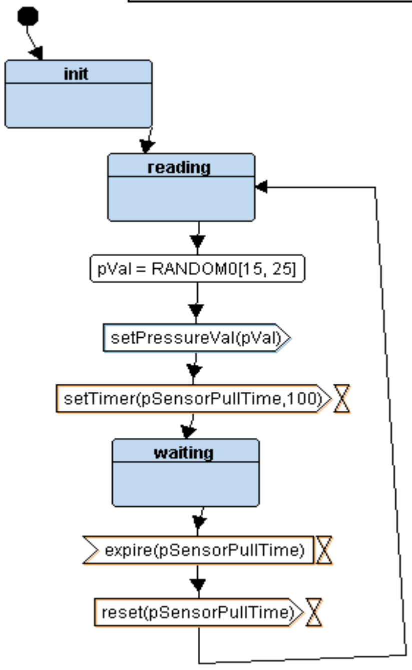
1-Pressure Sensor State Diagram:

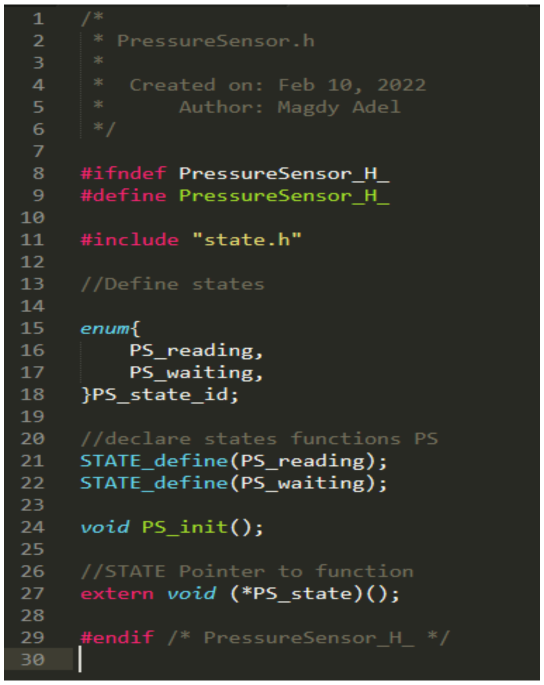


```c
 */
#include "PressureSensor.h"
#include "driver.h"

//variables
static unsigned int PS_pVal;

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

void PS_init()
{
	//init Pressure Sensor
	PS_state=STATE(PS_reading);
}


STATE_define(PS_reading)
{
	//state Name
	PS_state_id = PS_reading;

	//state Action
	//set Pressure Value
	PS_pVal = getPressureVal();

	PS_state=STATE(PS_waiting);
}

STATE_define(PS_waiting)
{
	//state Name
	PS_state_id = PS_waiting;

	//state Action
	Delay(60);
	//reset timer

	PS_state=STATE(PS_reading);
}
unsigned int SetPressureVal(void)
{
	return PS_pVal;
}
```

PressureSensor.c

PressureSensor

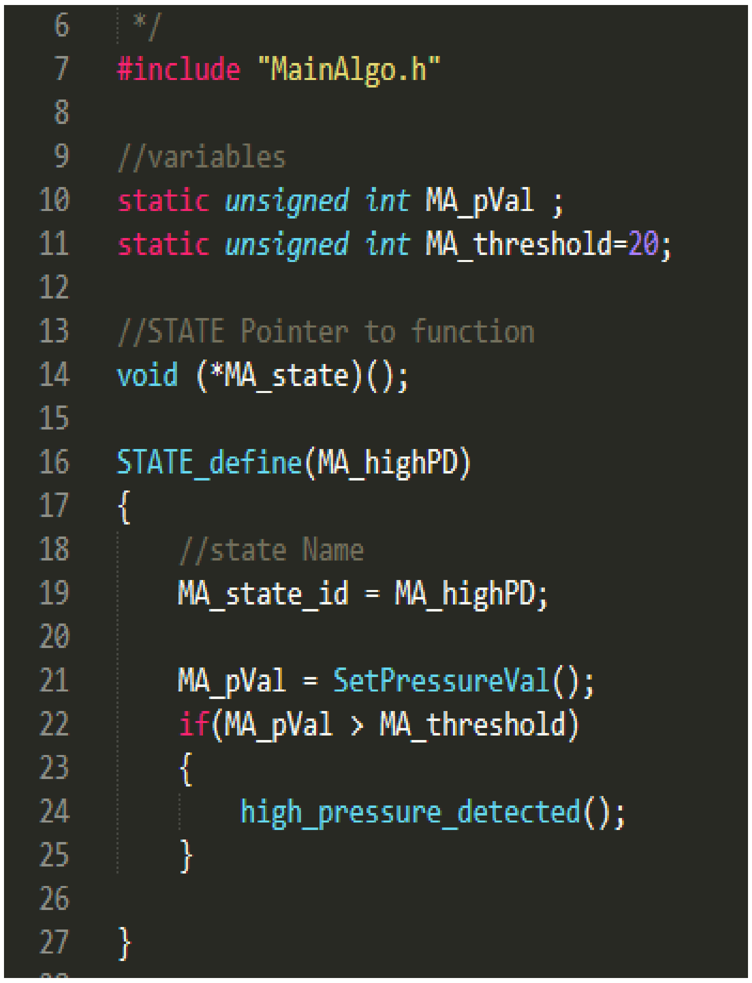




```c
/*
 * PressureSensor.h
 *
 *  Created on: Feb 10, 2022
 *      Author: Magdy Adel
 */

#ifndef PressureSensor_H_
#define PressureSensor_H_

#include "state.h"

//Define states

enum{
	PS_reading,
	PS_waiting,
}PS_state_id;

//declare states functions PS
STATE_define(PS_reading);
STATE_define(PS_waiting);

void PS_init();

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

#endif /* PressureSensor_H_ */
```

PressureSensor.h

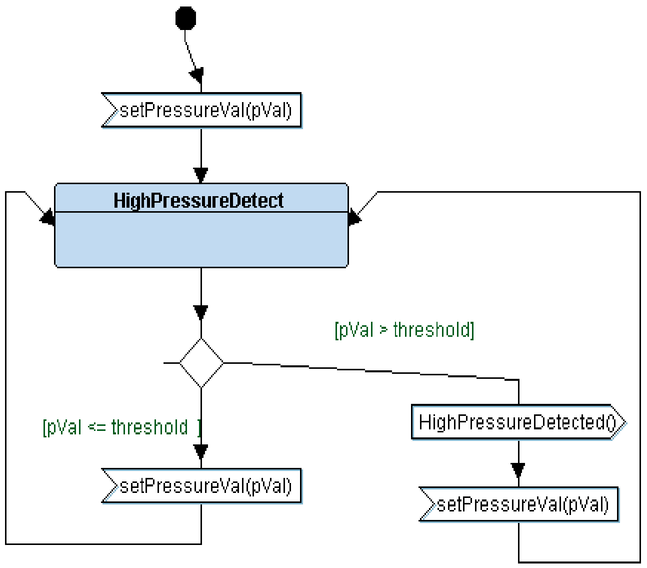
2-Main Algorithm State Diagram:

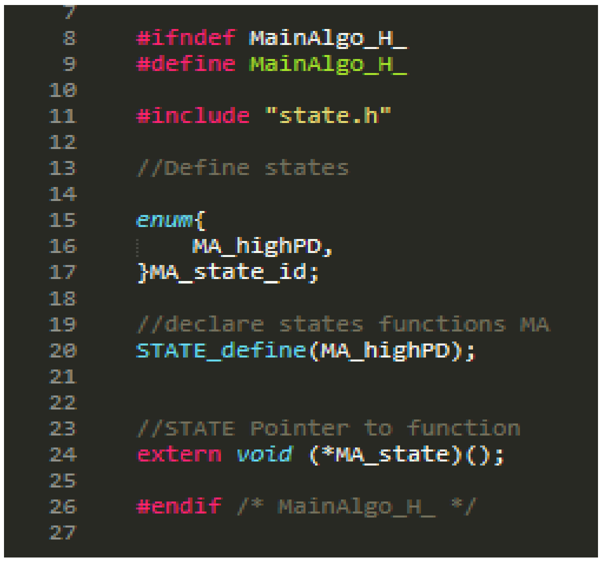


```
 */
#include "MainAlgo.h"

//variables
static unsigned int MA_pVal ;
static unsigned int MA_threshold=20;

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

STATE_define(MA_highPD)
{
    //state Name
    MA_state_id = MA_highPD;

    MA_pVal = SetPressureVal();
    if(MA_pVal > MA_threshold)
    {
        high_pressure_detected();
    }

}
```

MainAlgo.c

MainAlgorithm

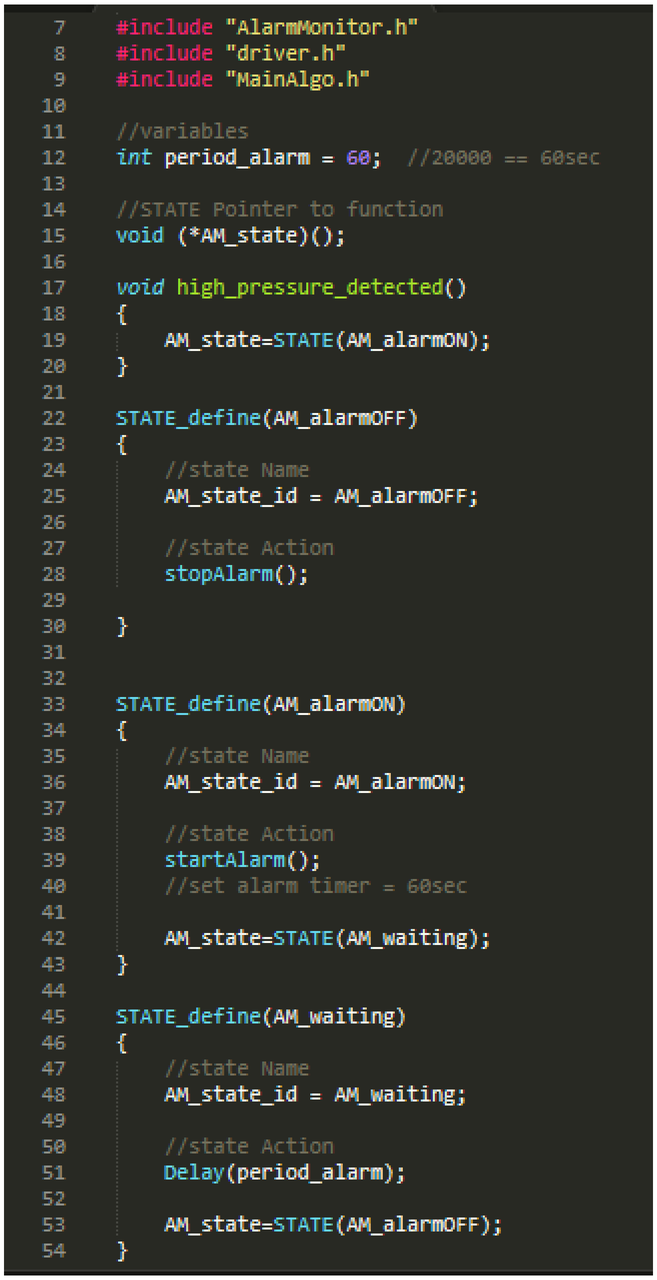




```
#ifndef MainAlgo_H_
#define MainAlgo_H_

#include "state.h"

//Define states

enum{
    MA_highPD,
}MA_state_id;

//declare states functions MA
STATE_define(MA_highPD);


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

#endif /* MainAlgo_H_ */
```

MainAlgo.h

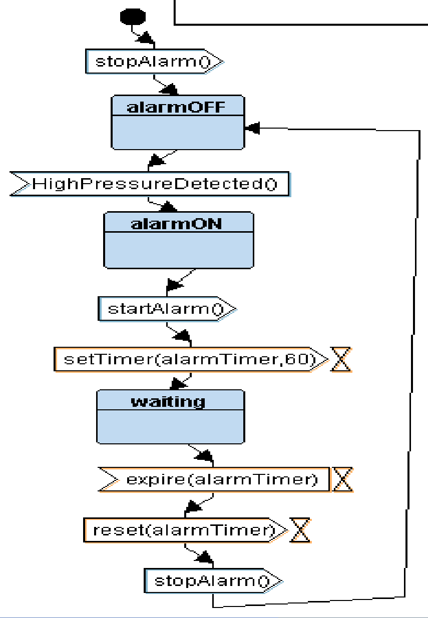
3-Alarm Monitor State Diagram:

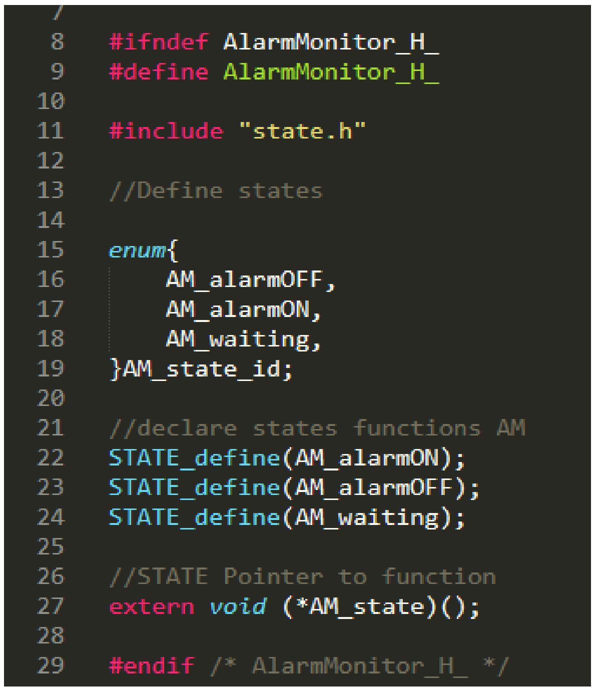


```c
#include "AlarmMonitor.h"
#include "driver.h"
#include "MainAlgo.h"

//variables
int period_alarm = 60;  //20000 == 60sec

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

void high_pressure_detected()
{
    AM_state=STATE(AM_alarmON);
}

STATE_define(AM_alarmOFF)
{
    //state Name
    AM_state_id = AM_alarmOFF;

    //state Action
    stopAlarm();

}


STATE_define(AM_alarmON)
{
    //state Name
    AM_state_id = AM_alarmON;

    //state Action
    startAlarm();
    //set alarm timer = 60sec

    AM_state=STATE(AM_waiting);
}

STATE_define(AM_waiting)
{
    //state Name
    AM_state_id = AM_waiting;

    //state Action
    Delay(period_alarm);

    AM_state=STATE(AM_alarmOFF);
}
```

AlarmMonitor.c

AlarmMonitor

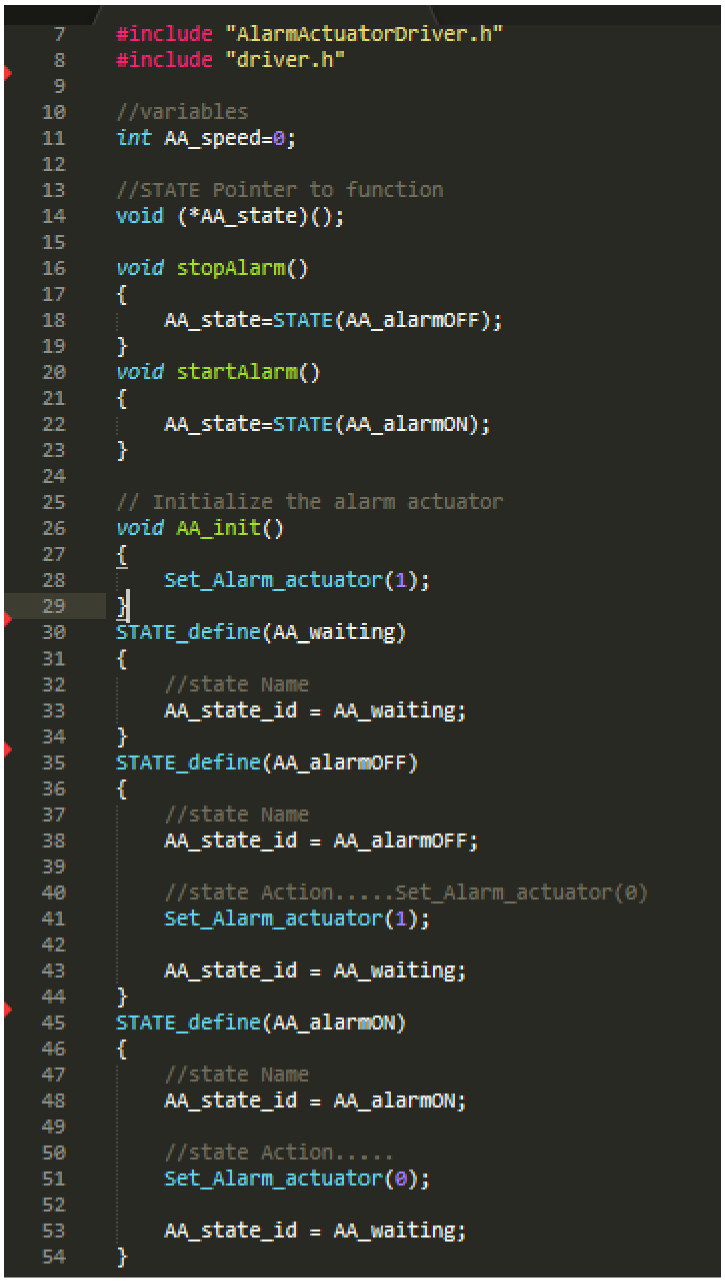




```c
#ifndef AlarmMonitor_H_
#define AlarmMonitor_H_

#include "state.h"

//Define states

enum{
    AM_alarmOFF,
    AM_alarmON,
    AM_waiting,
}AM_state_id;

//declare states functions AM
STATE_define(AM_alarmON);
STATE_define(AM_alarmOFF);
STATE_define(AM_waiting);

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

#endif /* AlarmMonitor_H_ */
```

AlarmMonitor.h

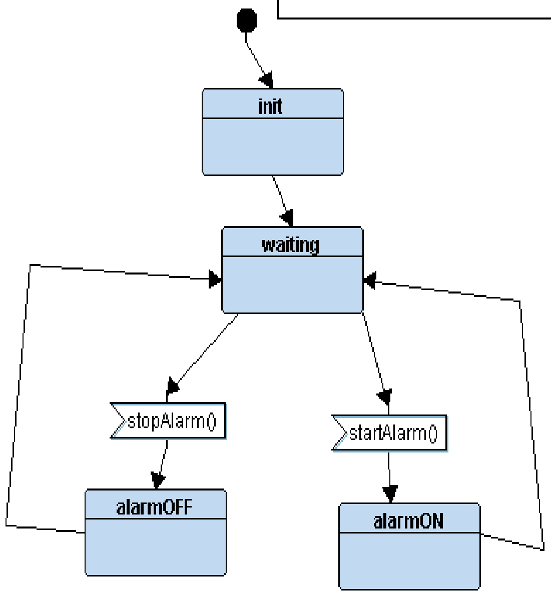
4-Alarm Actuator State Diagram:

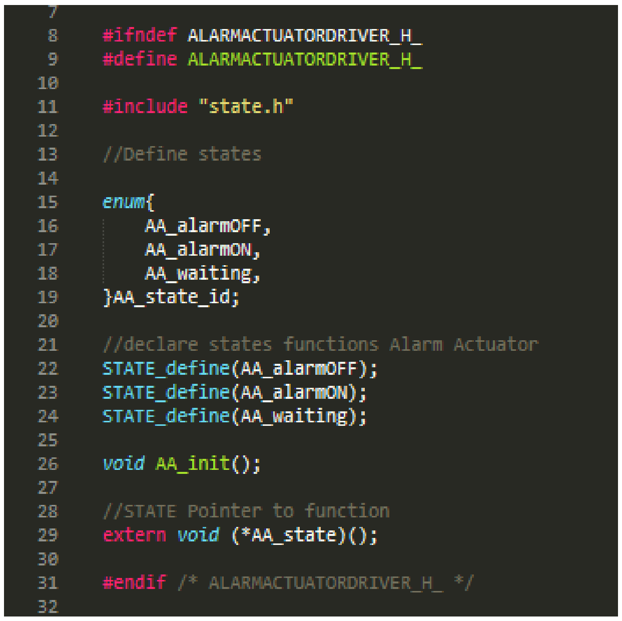


```c
#include "AlarmActuatorDriver.h"
#include "driver.h"

//variables
int AA_speed=0;

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

void stopAlarm()
{
    AA_state=STATE(AA_alarmOFF);
}
void startAlarm()
{
    AA_state=STATE(AA_alarmON);
}

// Initialize the alarm actuator
void AA_init()
{
    Set_Alarm_actuator(1);
}
STATE_define(AA_waiting)
{
    //state Name
    AA_state_id = AA_waiting;
}
STATE_define(AA_alarmOFF)
{
    //state Name
    AA_state_id = AA_alarmOFF;

    //state Action.....Set_Alarm_actuator(0)
    Set_Alarm_actuator(1);

    AA_state_id = AA_waiting;
}
STATE_define(AA_alarmON)
{
    //state Name
    AA_state_id = AA_alarmON;

    //state Action.....
    Set_Alarm_actuator(0);

    AA_state_id = AA_waiting;
}
```

AlarmActuator.c

AlarmActuator



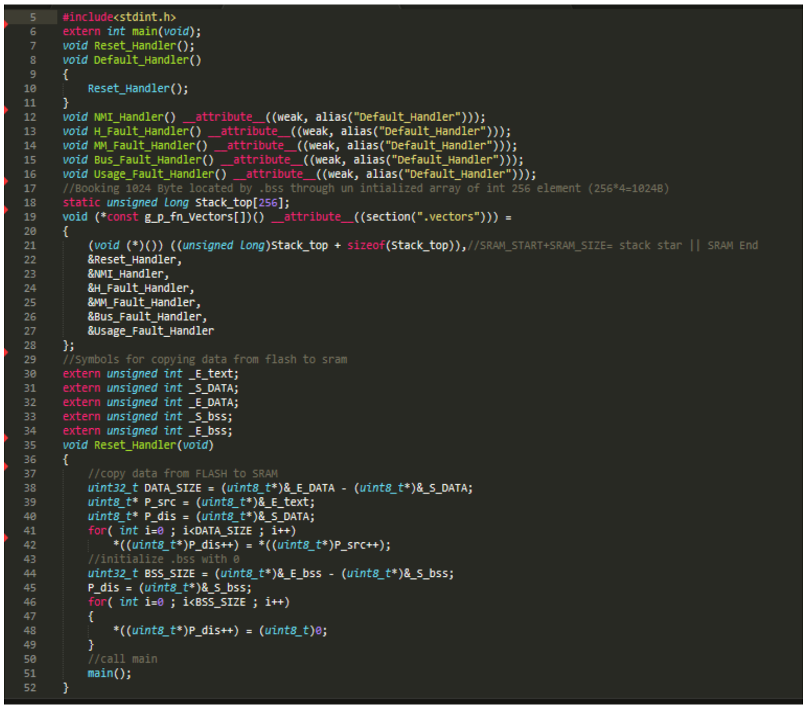


```c
#ifndef ALARMACTUATORDRIVER_H_
#define ALARMACTUATORDRIVER_H_

#include "state.h"

//Define states

enum{
    AA_alarmOFF,
    AA_alarmON,
    AA_waiting,
}AA_state_id;

//declare states functions Alarm Actuator
STATE_define(AA_alarmOFF);
STATE_define(AA_alarmON);
STATE_define(AA_waiting);

void AA_init();

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

#endif /* ALARMACTUATORDRIVER_H_ */
```

AlarmActuator.h

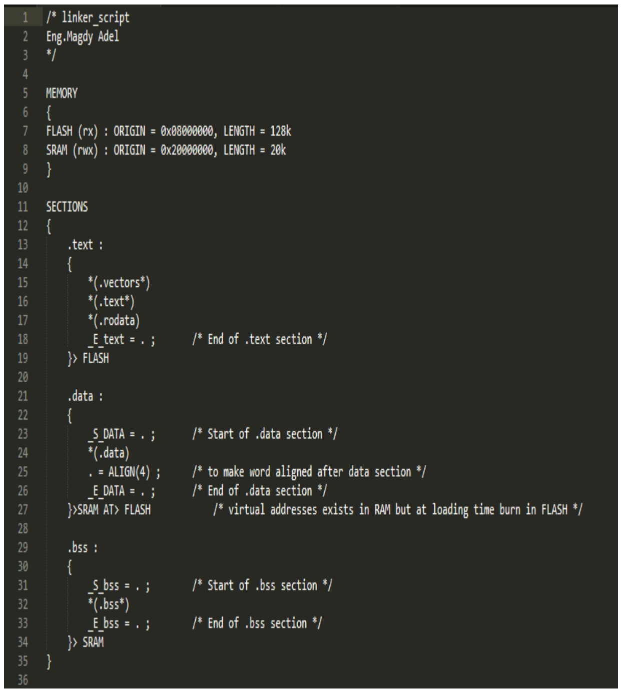
- Code

Startup.c



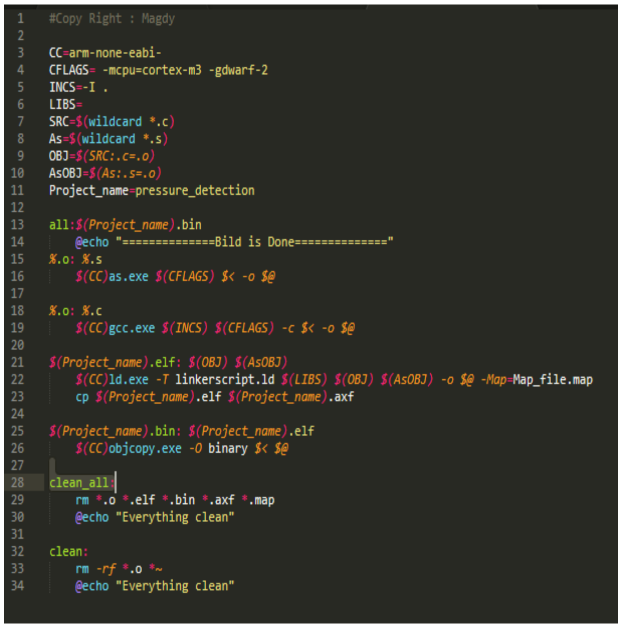
```c
#include<stdint.h>
extern int main(void);
void Reset_Handler();
void Default_Handler()
{
    Reset_Handler();
}
void NMI_Handler() __attribute__((weak, alias("Default_Handler")));
void H_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
void MM_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
void Bus_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
void Usage_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
//Booking 1024 Byte located by .bss through un intialized array of int 256 element (256*4=1024B)
static unsigned long Stack_top[256];
void (*const g_p_fn_Vectors[])() __attribute__((section(".vectors"))) =
{
    (void (*)()) ((unsigned long)Stack_top + sizeof(Stack_top)),//SRAM_START+SRAM_SIZE= stack star || SRAM End
    &Reset_Handler,
    &NMI_Handler,
    &H_Fault_Handler,
    &MM_Fault_Handler,
    &Bus_Fault_Handler,
    &Usage_Fault_Handler
};
//Symbols for copying data from flash to sram
extern unsigned int _E_text;
extern unsigned int _S_DATA;
extern unsigned int _E_DATA;
extern unsigned int _S_bss;
extern unsigned int _E_bss;
void Reset_Handler(void)
{
    //copy data from FLASH to SRAM
    uint32_t DATA_SIZE = (uint8_t*)&_E_DATA - (uint8_t*)&_S_DATA;
    uint8_t* P_src = (uint8_t*)&_E_text;
    uint8_t* P_dis = (uint8_t*)&_S_DATA;
    for( int i=0 ; i<DATA_SIZE ; i++)
        *((uint8_t*)P_dis++) = *((uint8_t*)P_src++);
    //initialize .bss with 0
    uint32_t BSS_SIZE = (uint8_t*)&_E_bss - (uint8_t*)&_S_bss;
    P_dis = (uint8_t*)&_S_bss;
    for( int i=0 ; i<BSS_SIZE ; i++)
    {
        *((uint8_t*)P_dis++) = (uint8_t)0;
    }
    //call main
    main();
}
```

Linkerscript.ld



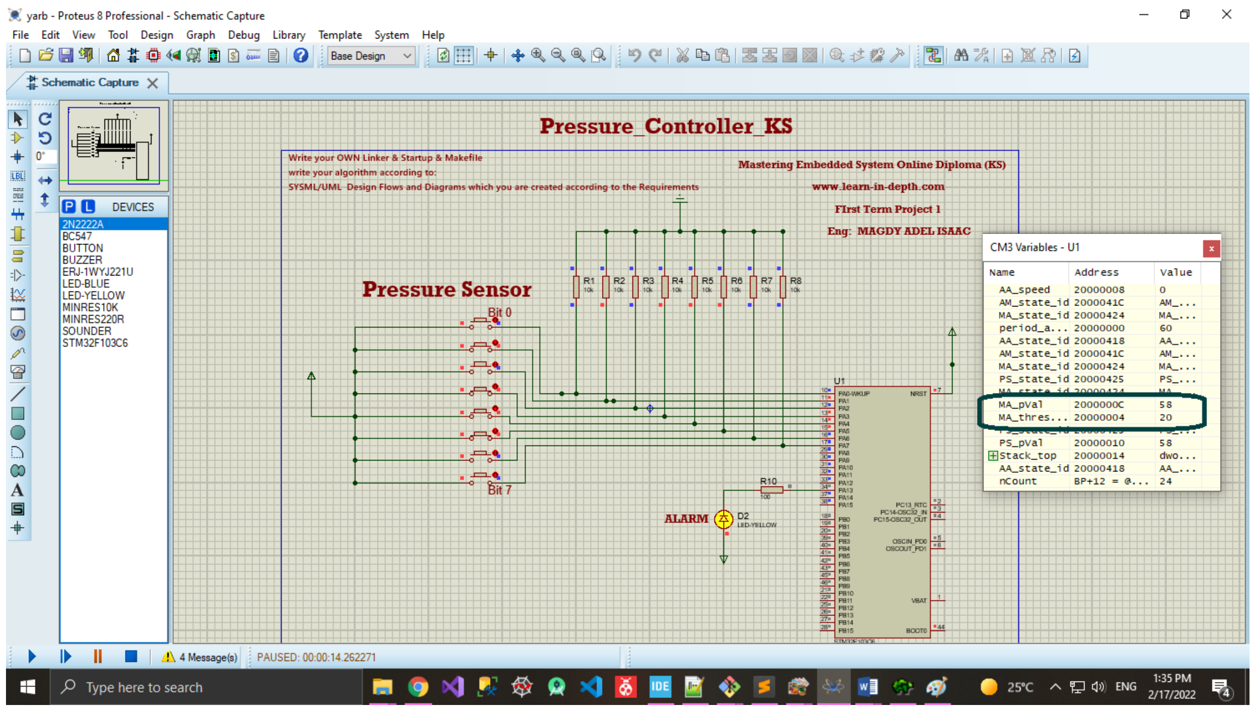
```
/* linker_script
Eng.Magdy Adel
*/

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

SECTIONS
{
    .text :
    {
        *(.vectors*)
        *(.text*)
        *(.rodata)
        _E_text = . ;       /* End of .text section */
    }> FLASH

    .data :
    {
        _S_DATA = . ;       /* Start of .data section */
        *(.data)
        . = ALIGN(4) ;      /* to make word aligned after data section */
        _E_DATA = . ;       /* End of .data section */
    }>SRAM AT> FLASH            /* virtual addresses exists in RAM but at loading time burn in FLASH */

    .bss :
    {
        _S_bss = . ;        /* Start of .bss section */
        *(.bss*)
        _E_bss = . ;        /* End of .bss section */
    }> SRAM
}

```

Makefile



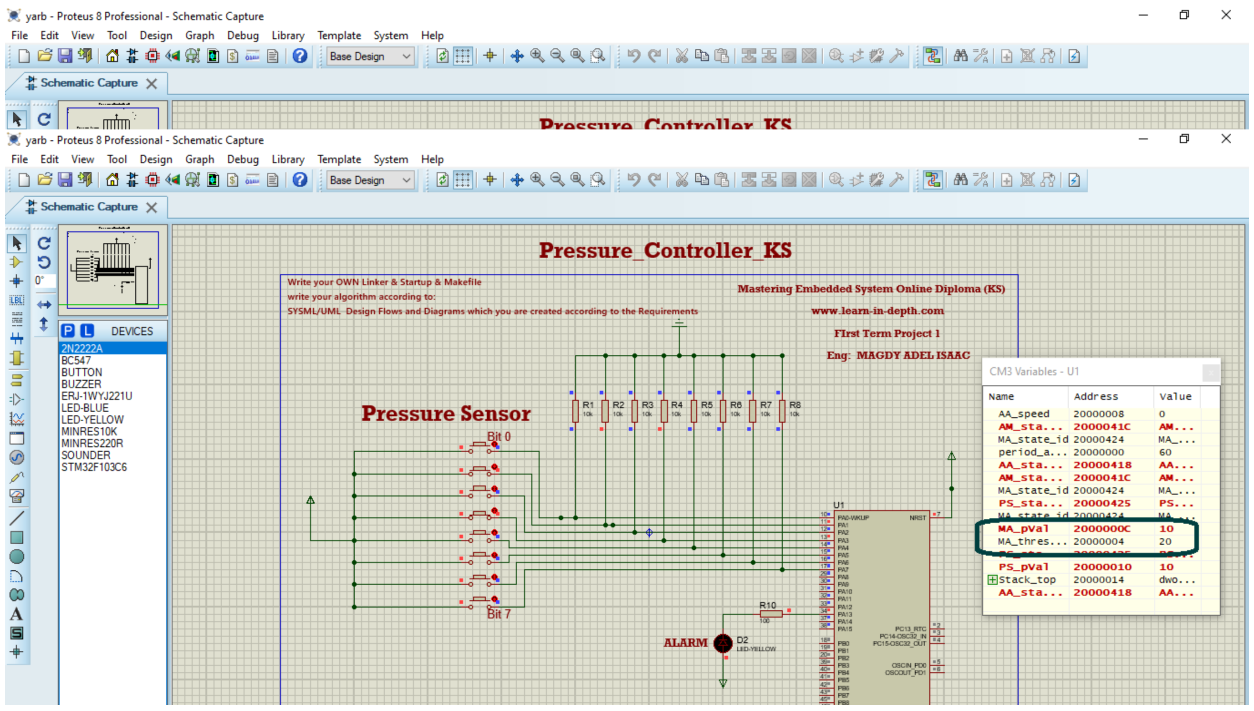
```makefile
#Copy Right : Magdy

CC=arm-none-eabi-
CFLAGS= -mcpu=cortex-m3 -gdwarf-2
INCS=-I .
LIBS=
SRC=$(wildcard *.c)
As=$(wildcard *.s)
OBJ=$(SRC:.c=.o)
AsOBJ=$(As:.s=.o)
Project_name=pressure_detection

all:$(Project_name).bin
	@echo "==============Bild is Done=============="
%.o: %.s
	$(CC)as.exe $(CFLAGS) $< -o $@

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

$(Project_name).elf: $(OBJ) $(AsOBJ)
	$(CC)ld.exe -T linkerscript.ld $(LIBS) $(OBJ) $(AsOBJ) -o $@ -Map=Map_file.map
	cp $(Project_name).elf $(Project_name).axf

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

clean_all:
	rm *.o *.elf *.bin *.axf *.map
	@echo "Everything clean"

clean:
	rm -rf *.o *~
	@echo "Everything clean"
```

- Proteus Simulation


yarb - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
DEVICES
2N2222A
BC547
BUTTON
BUZZER
ERJ-1WYJ221U
LED-BLUE
LED-YELLOW
MINRES10K
MINRES220R
SOUNDER
STM32F103C6
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: MAGDY ADEL ISAAC
Pressure Sensor
Bit 0
Bit 7
R1 R2 R3 R4 R5 R6 R7 R8
R10
ALARM
D2
LED-YELLOW
U1
4 Message(s)
PAUSED: 00:00:14.262271
Type here to search
25°C ENG 1:35 PM 2/17/2022

CM3 Variables - U1

Name	Address	Value
AA_speed	20000008	0
AM_state_id	2000041C	AM_...
MA_state_id	20000424	MA_...
period_a...	20000000	60
AA_state_id	20000418	AA_...
AM_state_id	2000041C	AM_...
MA_state_id	20000424	MA_...
PS_state_id	20000425	PS_...
MA_state_id	20000424	MA_...
MA_pVal	2000000C	58
MA_thres...	20000004	20
PS_state_id	20000425	PS_...
PS_pVal	20000010	58
⊞Stack_top	20000014	dwo...
AA_state_id	20000418	AA_...
nCount	BP+12 = @...	24

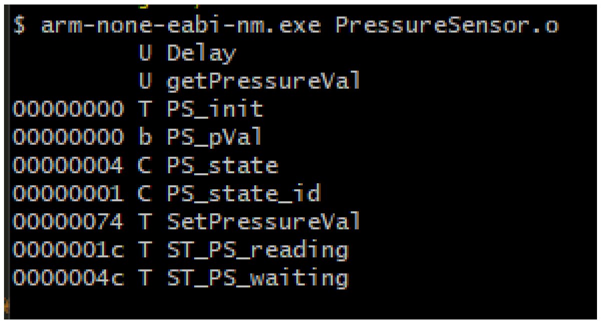

yarb - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
DEVICES
2N2222A
BC547
BUTTON
BUZZER
ERJ-1WYJ221U
LED-BLUE
LED-YELLOW
MINRES10K
MINRES220R
SOUNDER
STM32F103C6
Pressure_Controller_KS
Write your OWN Linker & 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: MAGDY ADEL ISAAC
Pressure Sensor
Bit 0
Bit 7
R1 R2 R3 R4 R5 R6 R7 R8
R10
ALARM
D2
LED-YELLOW
U1

CM3 Variables - U1

Name	Address	Value
AA_speed	20000008	0
AM_sta...	2000041C	AM...
MA_state_id	20000424	MA_...
period_a...	20000000	60
AA_sta...	20000418	AA...
AM_sta...	2000041C	AM...
MA_state_id	20000424	MA_...
PS_sta...	20000425	PS...
MA_state_id	20000424	MA_...
MA_pVal	2000000C	10
MA_thres...	20000004	20
PS_sta...	20000425	PS...
PS_pVal	20000010	10
⊞Stack_top	20000014	dwo...
AA_sta...	20000418	AA...

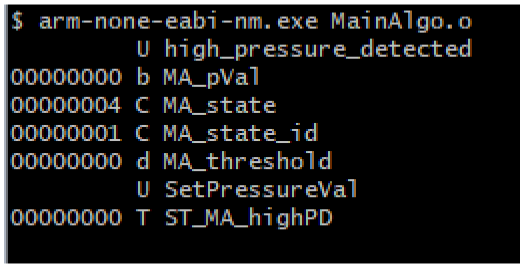
- Symbols

PressureSensor.o



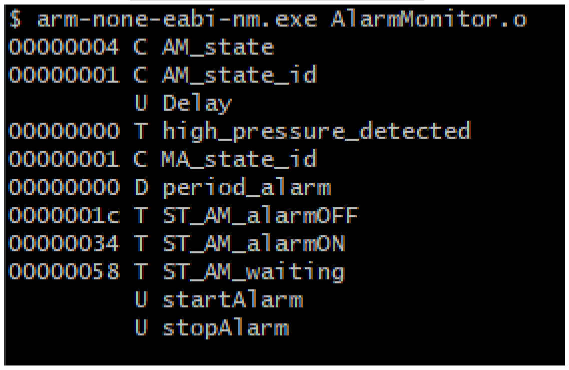
```
$ arm-none-eabi-nm.exe PressureSensor.o
         U Delay
         U getPressureVal
00000000 T PS_init
00000000 b PS_pVal
00000004 C PS_state
00000001 C PS_state_id
00000074 T SetPressureVal
0000001c T ST_PS_reading
0000004c T ST_PS_waiting
```

MainAlgo.o



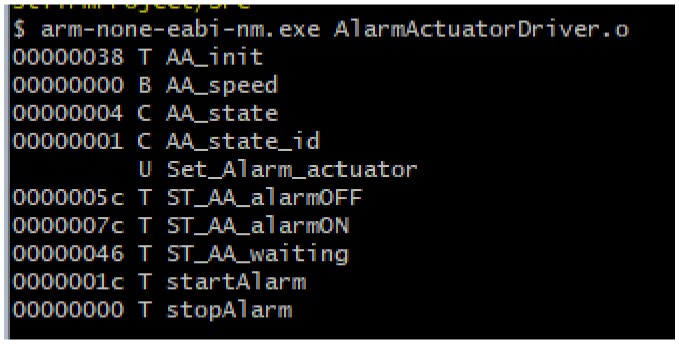
```
$ arm-none-eabi-nm.exe MainAlgo.o
         U high_pressure_detected
00000000 b MA_pVal
00000004 C MA_state
00000001 C MA_state_id
00000000 d MA_threshold
         U SetPressureVal
00000000 T ST_MA_highPD
```

AlarmMonitor.o



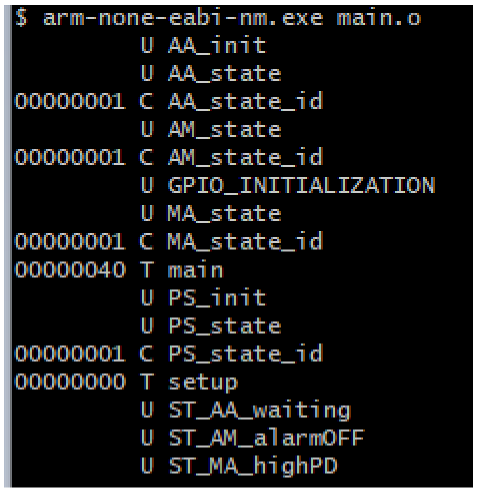
```
$ arm-none-eabi-nm.exe AlarmMonitor.o
00000004 C AM_state
00000001 C AM_state_id
         U Delay
00000000 T high_pressure_detected
00000001 C MA_state_id
00000000 D period_alarm
0000001c T ST_AM_alarmOFF
00000034 T ST_AM_alarmON
00000058 T ST_AM_waiting
         U startAlarm
         U stopAlarm
```

AlarmActuatorDriver.o



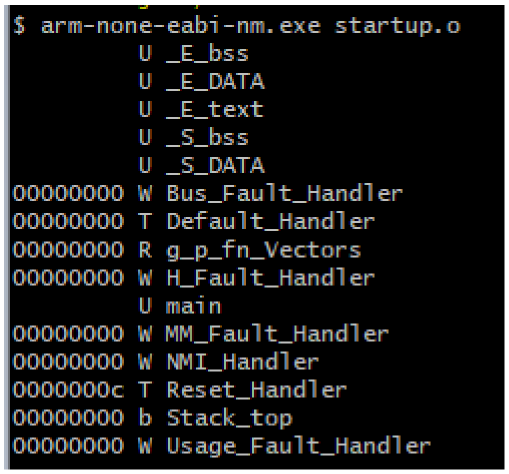
```
$ arm-none-eabi-nm.exe AlarmActuatorDriver.o
00000038 T AA_init
00000000 B AA_speed
00000004 C AA_state
00000001 C AA_state_id
         U Set_Alarm_actuator
0000005c T ST_AA_alarmOFF
0000007c T ST_AA_alarmON
00000046 T ST_AA_waiting
0000001c T startAlarm
00000000 T stopAlarm
```

Main.o



```
$ arm-none-eabi-nm.exe main.o
         U AA_init
         U AA_state
00000001 C AA_state_id
         U AM_state
00000001 C AM_state_id
         U GPIO_INITIALIZATION
         U MA_state
00000001 C MA_state_id
00000040 T main
         U PS_init
         U PS_state
00000001 C PS_state_id
00000000 T setup
         U ST_AA_waiting
         U ST_AM_alarmOFF
         U ST_MA_highPD
```

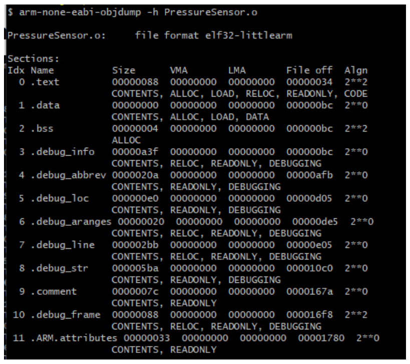
Startup.o



```
$ arm-none-eabi-nm.exe startup.o
         U _E_bss
         U _E_DATA
         U _E_text
         U _S_bss
         U _S_DATA
00000000 W Bus_Fault_Handler
00000000 T Default_Handler
00000000 R g_p_fn_Vectors
00000000 W H_Fault_Handler
         U main
00000000 W MM_Fault_Handler
00000000 W NMI_Handler
0000000c T Reset_Handler
00000000 b Stack_top
00000000 W Usage_Fault_Handler
```

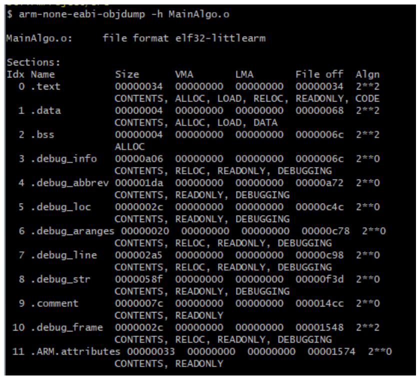
- Sections

PressureSensor.o



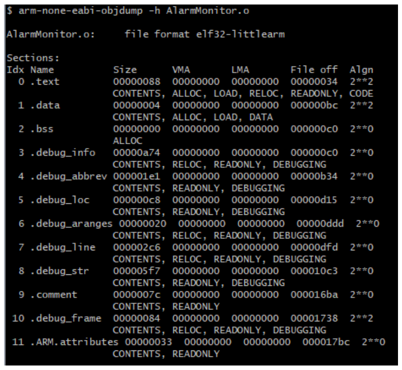
```
$ arm-none-eabi-objdump -h PressureSensor.o

PressureSensor.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   00000a3f  00000000  00000000  000000bc  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 0000020a  00000000  00000000  00000afb  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000000e0  00000000  00000000  00000d05  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000de5  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002bb  00000000  00000000  00000e05  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000005ba  00000000  00000000  000010c0  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  0000167a  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000088  00000000  00000000  000016f8  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001780  2**0
                  CONTENTS, READONLY
```

MainAlgo.o



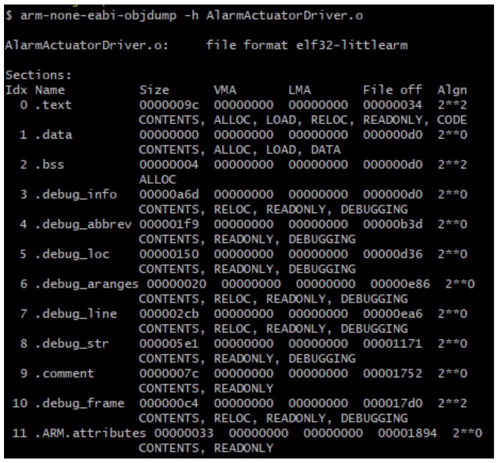
```
$ arm-none-eabi-objdump -h MainAlgo.o

MainAlgo.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000034  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000004  00000000  00000000  00000068  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  0000006c  2**2
                  ALLOC
  3 .debug_info   00000a06  00000000  00000000  0000006c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001da  00000000  00000000  00000a72  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000002c  00000000  00000000  00000c4c  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000c78  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002a5  00000000  00000000  00000c98  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000058f  00000000  00000000  00000f3d  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  000014cc  2**0
                  CONTENTS, READONLY
 10 .debug_frame  0000002c  00000000  00000000  00001548  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001574  2**0
                  CONTENTS, READONLY
```

AlarmMonitor.o



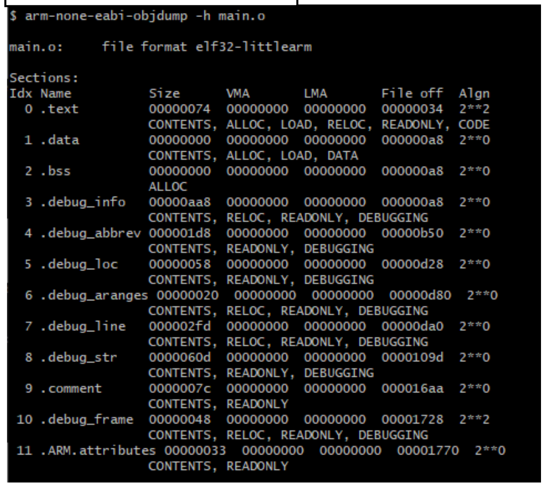
```
$ arm-none-eabi-objdump -h AlarmMonitor.o

AlarmMonitor.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         00000004  00000000  00000000  000000bc  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000c0  2**0
                  ALLOC
  3 .debug_info   00000a74  00000000  00000000  000000c0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001e1  00000000  00000000  00000b34  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000000c8  00000000  00000000  00000d15  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000ddd  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002c6  00000000  00000000  00000dfd  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000005f7  00000000  00000000  000010c3  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  000016ba  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000084  00000000  00000000  00001738  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  000017bc  2**0
                  CONTENTS, READONLY
```

AlarmActuatorDriver.o



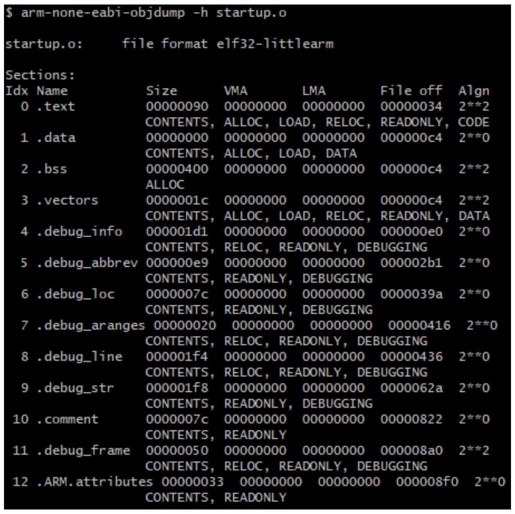
```
$ arm-none-eabi-objdump -h AlarmActuatorDriver.o

AlarmActuatorDriver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000009c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000d0  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  000000d0  2**2
                  ALLOC
  3 .debug_info   00000a6d  00000000  00000000  000000d0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001f9  00000000  00000000  00000b3d  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000150  00000000  00000000  00000d36  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000e86  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002cb  00000000  00000000  00000ea6  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000005e1  00000000  00000000  00001171  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  00001752  2**0
                  CONTENTS, READONLY
 10 .debug_frame  000000c4  00000000  00000000  000017d0  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001894  2**0
                  CONTENTS, READONLY
```

Main.o



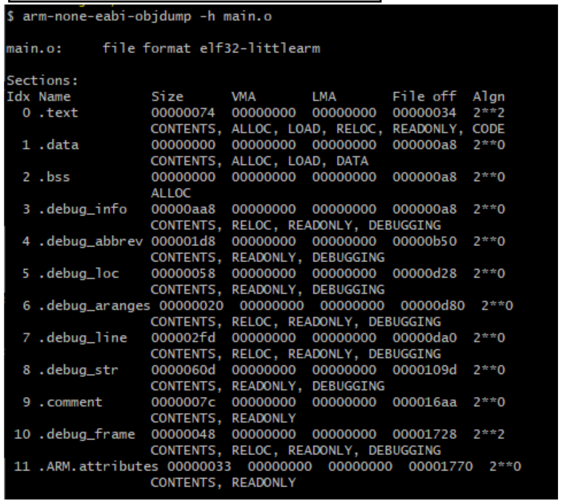
```
$ arm-none-eabi-objdump -h main.o

main.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         00000000  00000000  00000000  000000a8  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000a8  2**0
                  ALLOC
  3 .debug_info   00000aa8  00000000  00000000  000000a8  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001d8  00000000  00000000  00000b50  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000058  00000000  00000000  00000d28  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000d80  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002fd  00000000  00000000  00000da0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000060d  00000000  00000000  0000109d  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  000016aa  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000048  00000000  00000000  00001728  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001770  2**0
                  CONTENTS, READONLY
```

Startup.o



```
$ arm-none-eabi-objdump -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          00000400  00000000  00000000  000000c4  2**2
                  ALLOC
  3 .vectors      0000001c  00000000  00000000  000000c4  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, DATA
  4 .debug_info   000001d1  00000000  00000000  000000e0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 000000e9  00000000  00000000  000002b1  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    0000007c  00000000  00000000  0000039a  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  00000416  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_line   000001f4  00000000  00000000  00000436  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  9 .debug_str    000001f8  00000000  00000000  0000062a  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .comment      0000007c  00000000  00000000  00000822  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000050  00000000  00000000  000008a0  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 12 .ARM.attributes 00000033  00000000  00000000  000008f0  2**0
                  CONTENTS, READONLY
```

Pressure_detection.elf



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

main.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         00000000  00000000  00000000  000000a8  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000a8  2**0
                  ALLOC
  3 .debug_info   00000aa8  00000000  00000000  000000a8  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001d8  00000000  00000000  00000b50  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000058  00000000  00000000  00000d28  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000d80  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000002fd  00000000  00000000  00000da0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000060d  00000000  00000000  0000109d  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  000016aa  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000048  00000000  00000000  00001728  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001770  2**0
                  CONTENTS, READONLY
```