Mastering Embedded System Online Diploma

www.learn-in-depth.com

First Term (Final project 1)

Engineer. Mai Mahmoud Mousa

My profile:

learn-in-depth/online-diploma/maimousa994@gmail.com

High pressure detection system

System design sequence:

1. Case study:

The client expects the delivery of the software of the following system:

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

Assumptions:

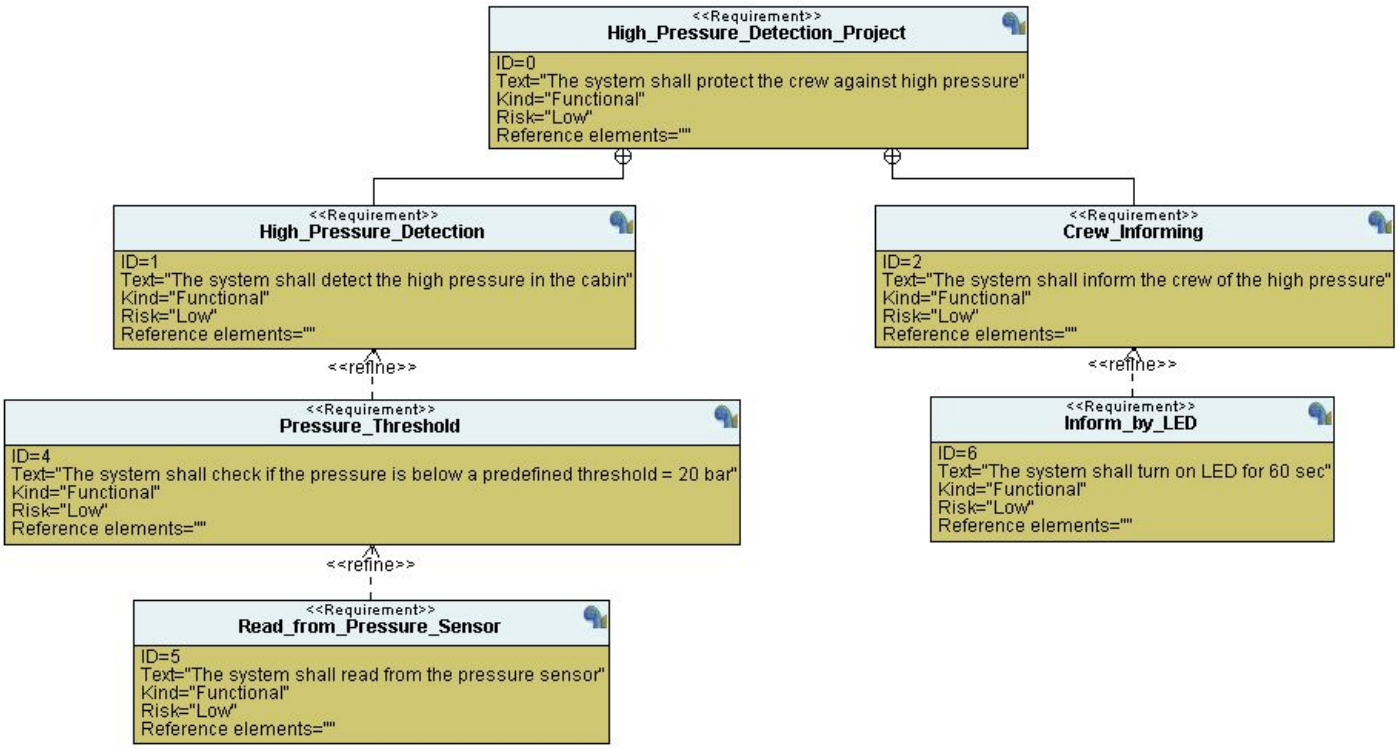
- The controller setup and shut down procedures are not modeled.
- The controller maintenance is not modeled.
- The pressure sensor never fails.
- The alarm never fails.
- The controller never faces power cut.

2. Method:

The chosen method in designing and implementing this system is the V-model method.

3. System requirements:

The following diagram is the UML requirement diagram for this system.



4. Space exploration and hardware / software partitioning:

Hardware:

- Controller: STM32F103C6.
- Alarm: LED.
- Sensor: Pressure sensor.

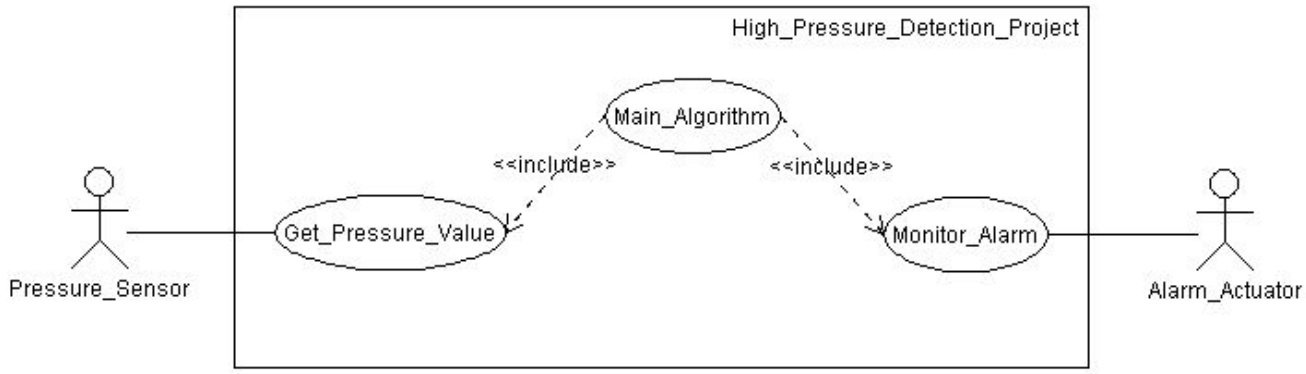
Software:

Here, we have four software modules:

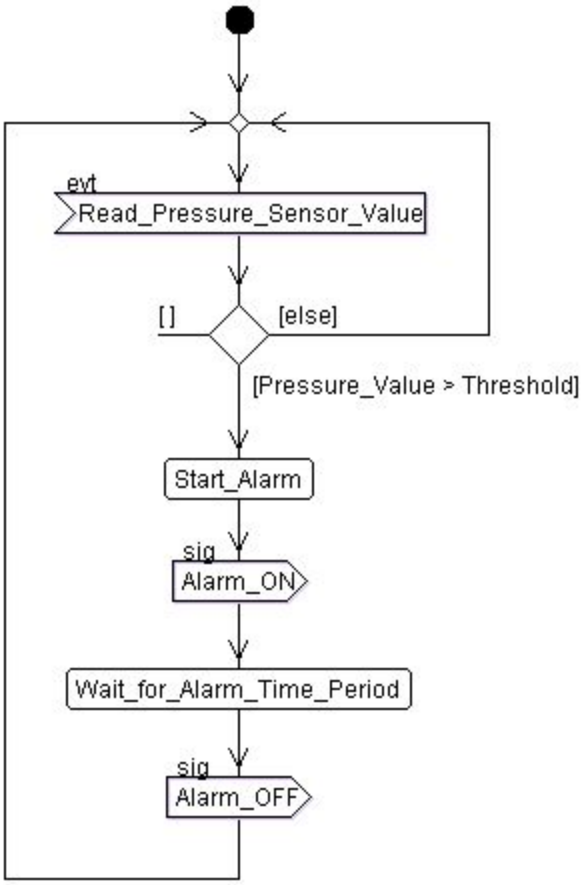
- Pressure sensor driver.
- Main algorithm.
- Alarm monitor.
- Alarm actuator driver.

5. System analysis:

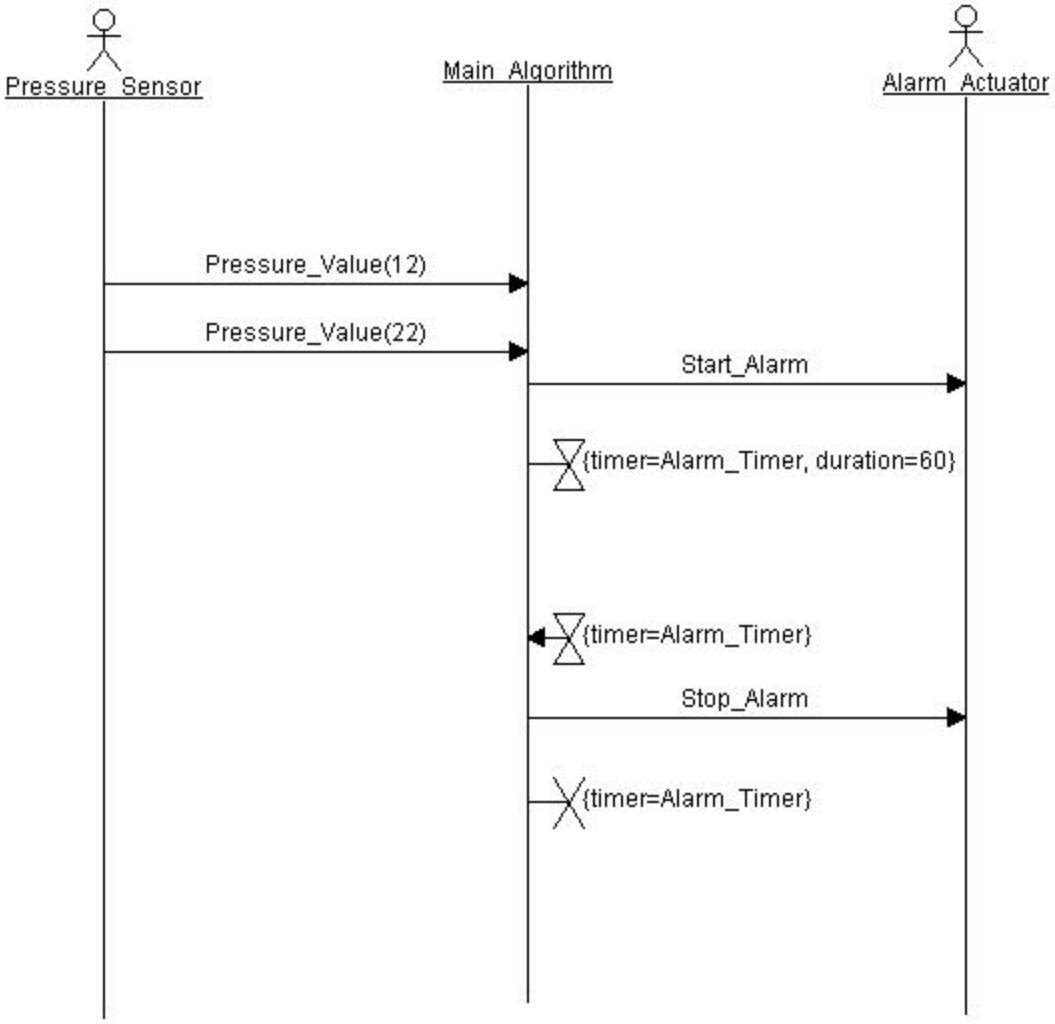
1. Use case diagram:



2. Activity diagram:

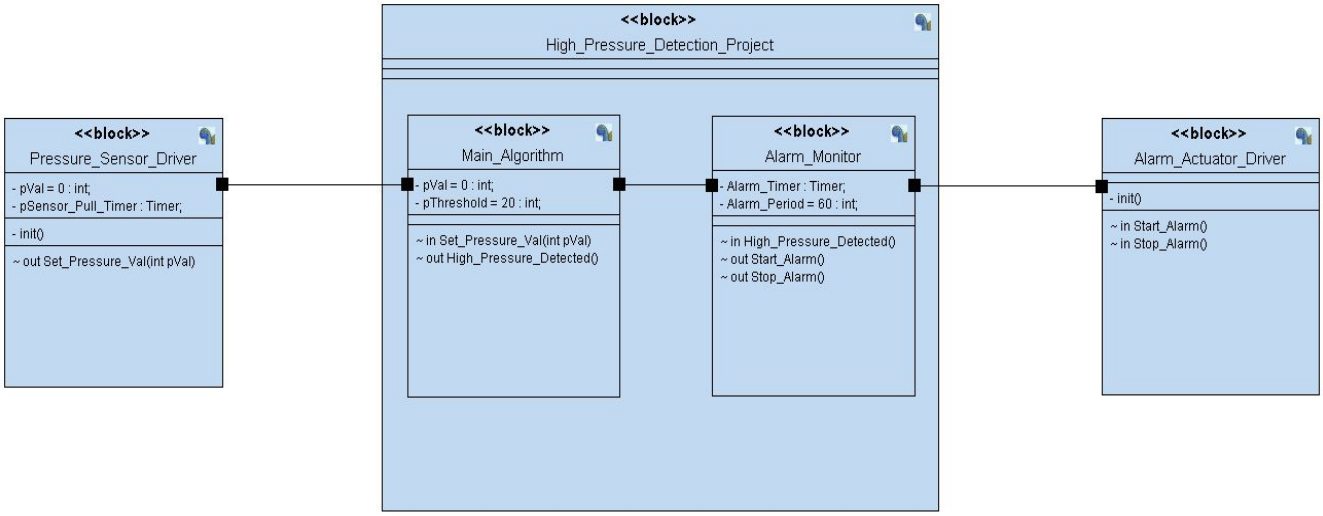


3. Sequence diagram:



6. System design:

System block diagram:



Software components state machines and implementation:

- **Pressure sensor driver:**

- Pressure_Sensor_Driver.c.

Pressure_Sensor_Driver.c

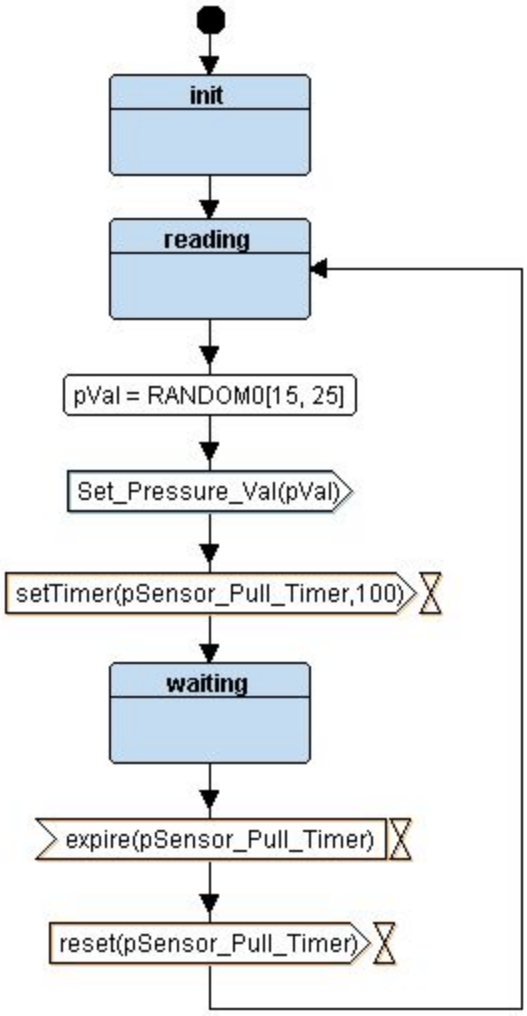
```c
/* Variables */
int pSensor_Driver_pVal = 0;

/* State pointer (pointer to function) */
void (*PRESSURE_SENSOR_DRIVER_state)();

/* Module Init */
void Pressure_Sensor_Driver_init (void) {   /* Driver Init */ }

/* State(s) implementation */
STATE_define(PRESSURE_SENSOR_DRIVER_reading)
{
    /* State name */
    PRESSURE_SENSOR_DRIVER_state_id = PRESSURE_SENSOR_DRIVER_reading;
    /* State action */
    /* 1. Get pressure reading from pressure sensor */
    pSensor_Driver_pVal = getPressureVal();
    /* 2. Send pressure sensor reading to the main algorithm */
    Set_Pressure_Val (pSensor_Driver_pVal);
    /* 3. Set pressure sensor pulling timer for 100 ms */
    Delay(100);
    /* Set next state */
    PRESSURE_SENSOR_DRIVER_state = STATE(PRESSURE_SENSOR_DRIVER_waiting);
}

STATE_define(PRESSURE_SENSOR_DRIVER_waiting)
{
    /* State name */
    PRESSURE_SENSOR_DRIVER_state_id = PRESSURE_SENSOR_DRIVER_waiting;
    /* State action: No action */
    /* Set next state */
    PRESSURE_SENSOR_DRIVER_state = STATE(PRESSURE_SENSOR_DRIVER_reading);
}
```

- Pressure_Sensor_Driver.h.

Pressure_Sensor_Driver.h

```c
#ifndef PRESSURE_SENSOR_DRIVER_
#define PRESSURE_SENSOR_DRIVER_

#include "state.h"

/* Define states */
enum
{
    PRESSURE_SENSOR_DRIVER_reading,
    PRESSURE_SENSOR_DRIVER_waiting
} PRESSURE_SENSOR_DRIVER_state_id;

/* Declare init function */
void Pressure_Sensor_Driver_init (void);

/* Declare states functions */
STATE_define(PRESSURE_SENSOR_DRIVER_reading);
STATE_define(PRESSURE_SENSOR_DRIVER_waiting);

/* State pointer (pointer to function) */
extern void (*PRESSURE_SENSOR_DRIVER_state)();

#endif
```

- Pressure sensor driver state machine.



- **Main algorithm:**

- Main_Algorithm.c.

Main_Algorithm.c

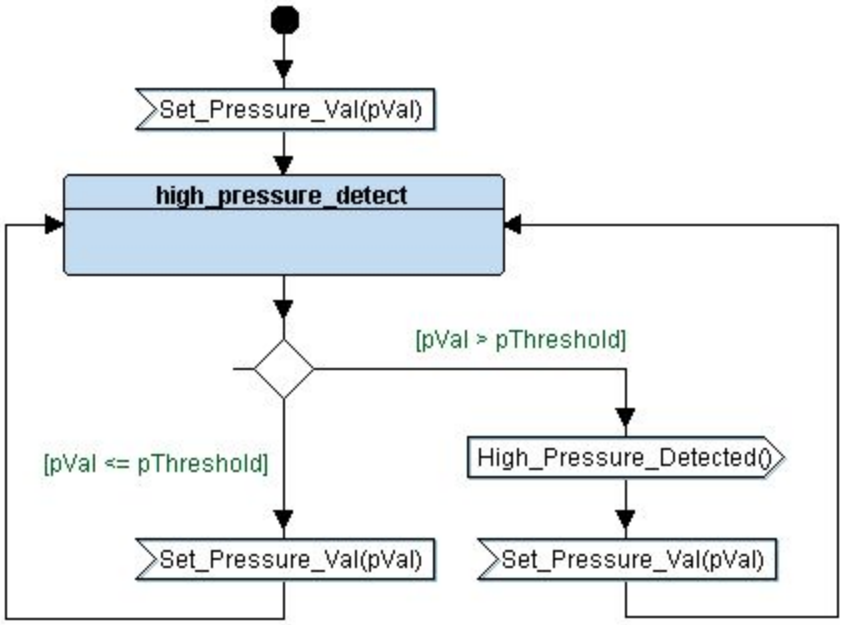
```c
#include "Main_Algorithm.h"
#include "state.h"


/* Variables */
int Main_Algorithm_pVal = 0;
int pThreshold = 20;

/* State pointer (pointer to function) */
void (*MAIN_ALGORITHM_state)();

/* Input signal(s) implementation */
void Set_Pressure_Val (int pVal)
{
    Main_Algorithm_pVal = pVal;
    /* Set next state */
    MAIN_ALGORITHM_state = STATE(MAIN_ALGORITHM_high_pressure_detect);
}

/* State(s) implementation */
STATE_define(MAIN_ALGORITHM_high_pressure_detect)
{
    /* State name */
    MAIN_ALGORITHM_state_id = MAIN_ALGORITHM_high_pressure_detect;
    /* State action */
    if      (Main_Algorithm_pVal <= pThreshold) { }
    else if (Main_Algorithm_pVal >  pThreshold) { High_Pressure_Detected(); }
}
```

- Main_Algorithm.h.

Main_Algorithm.h

```c
#ifndef MAIN_ALGORITHM_
#define MAIN_ALGORITHM_

#include "state.h"

/* Define states */
enum
{
    MAIN_ALGORITHM_high_pressure_detect
} MAIN_ALGORITHM_state_id;

/* Declare states functions */
STATE_define(MAIN_ALGORITHM_high_pressure_detect);

/* State pointer (pointer to function) */
extern void (*MAIN_ALGORITHM_state)();

#endif
```

- Main algorithm state machine.



- **Alarm monitor:**

- Alarm_Monitor.c.

Alarm_Monitor.c

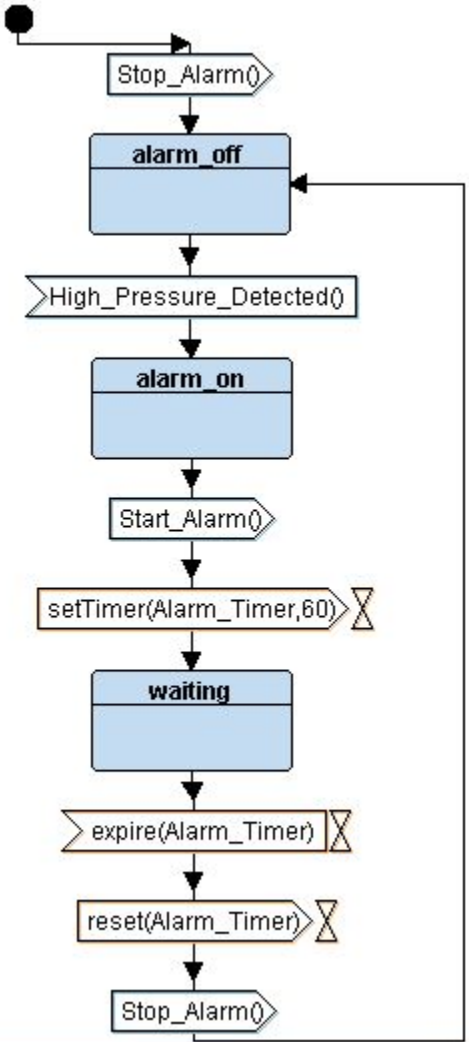
```c
/* Variables */
int Alarm_Period = 60;
/* State pointer (pointer to function) */
void (*ALARM_MONITOR_state)();
/* Input signal(s) implementation */
void High_Pressure_Detected (void)
{   ALARM_MONITOR_state = STATE(ALARM_MONITOR_alarm_on);  /* Set the next state */   }
/* State(s) implementation */
STATE_define(ALARM_MONITOR_alarm_off)
{   ALARM_MONITOR_state_id = ALARM_MONITOR_alarm_off;  /* State name */ /* State action: No action */  }
STATE_define(ALARM_MONITOR_alarm_on)
{
    /* State name */
    ALARM_MONITOR_state_id = ALARM_MONITOR_alarm_on;
    /* State action */
    /* 1. Send start alarm signal to Alarm_Actuator_Driver */
    Start_Alarm ();
    /* 2. Set a timer with the required alarm period (60 Sec) */
    Delay(Alarm_Period);
    /* Set next state */
    ALARM_MONITOR_state = STATE(ALARM_MONITOR_waiting);
}
STATE_define(ALARM_MONITOR_waiting)
{
    /* State name */
    ALARM_MONITOR_state_id = ALARM_MONITOR_waiting;
    /* State action */
    /* 1. Send stop alarm signal to Alarm_Actuator_Driver */
    Stop_Alarm ();
    /* Set next state */
    ALARM_MONITOR_state = STATE(ALARM_MONITOR_alarm_off);
}
```

- Alarm_Monitor.h.

Alarm_Monitor.h

```c
#ifndef ALARM_MONITOR_
#define ALARM_MONITOR_

#include "state.h"

/* Define states */
enum
{
    ALARM_MONITOR_alarm_off,
    ALARM_MONITOR_alarm_on,
    ALARM_MONITOR_waiting
} ALARM_MONITOR_state_id;

/* Declare states functions */
STATE_define(ALARM_MONITOR_alarm_off);
STATE_define(ALARM_MONITOR_alarm_on);
STATE_define(ALARM_MONITOR_waiting);

/* State pointer (pointer to function) */
extern void (*ALARM_MONITOR_state)();

#endif
```

- Alarm monitor state machine.



- **Alarm actuator driver:**
- Alarm_Actuator_Driver.c.

Alarm_Actuator_Driver.c

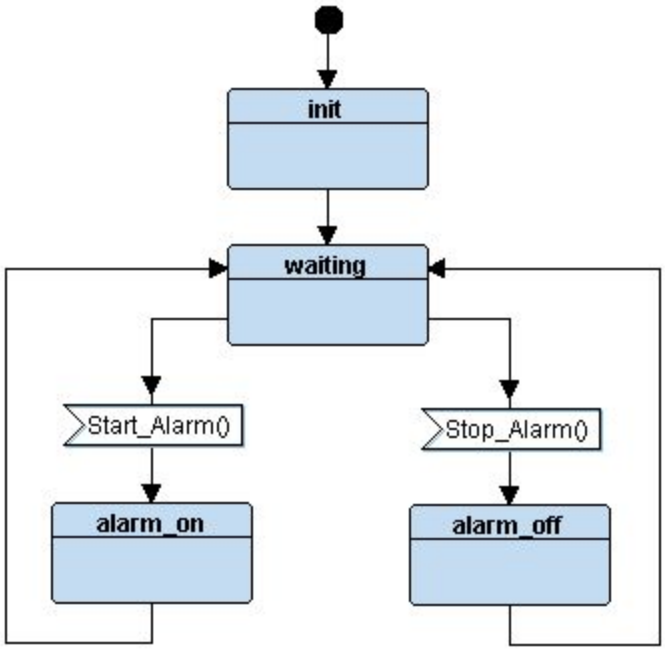
```c
/* Variables */
/* State pointer (pointer to function) */
void (*ALARM_ACTUATOR_DRIVER_state)();

/* Module Init */
void Alarm_Actuator_Driver_init (void)  {   /* Driver Init */   }

/* Input signal(s) implementation */
void Start_Alarm (void)
{   ALARM_ACTUATOR_DRIVER_state = STATE(ALARM_ACTUATOR_DRIVER_alarm_on);    /* Set the next state */   }
void Stop_Alarm (void)
{   ALARM_ACTUATOR_DRIVER_state = STATE(ALARM_ACTUATOR_DRIVER_alarm_off);  /* Set the next state */    }

/* State(s) implementation */
STATE_define(ALARM_ACTUATOR_DRIVER_alarm_off)
{
    ALARM_ACTUATOR_DRIVER_state_id = ALARM_ACTUATOR_DRIVER_alarm_off;   /* State name */
    /* State action */
    Set_Alarm_actuator(1);  /* Send signal to turn off the alarm */
    ALARM_ACTUATOR_DRIVER_state = STATE(ALARM_ACTUATOR_DRIVER_waiting); /* Set next state */
}

STATE_define(ALARM_ACTUATOR_DRIVER_alarm_on)
{
    ALARM_ACTUATOR_DRIVER_state_id = ALARM_ACTUATOR_DRIVER_alarm_on;    /* State name */
    /* State action */
    Set_Alarm_actuator(0);  /* Send signal to turn on the alarm */
    ALARM_ACTUATOR_DRIVER_state = STATE(ALARM_ACTUATOR_DRIVER_waiting); /* Set next state */
}

STATE_define(ALARM_ACTUATOR_DRIVER_waiting)
{
    ALARM_ACTUATOR_DRIVER_state_id = ALARM_ACTUATOR_DRIVER_waiting; /* State name */
    /* State action: No action */
}
```

- Alarm_Actuator_Driver.h.

Alarm_Actuator_Driver.h

```h
#ifndef ALARM_ACTUATOR_DRIVER_
#define ALARM_ACTUATOR_DRIVER_

#include "state.h"

/* Define states */
enum
{
    ALARM_ACTUATOR_DRIVER_alarm_off,
    ALARM_ACTUATOR_DRIVER_alarm_on,
    ALARM_ACTUATOR_DRIVER_waiting
} ALARM_ACTUATOR_DRIVER_state_id;

/* Declare init function */
void Alarm_Actuator_Driver_init (void);

/* Declare states functions */
STATE_define(ALARM_ACTUATOR_DRIVER_alarm_off);
STATE_define(ALARM_ACTUATOR_DRIVER_alarm_on);
STATE_define(ALARM_ACTUATOR_DRIVER_waiting);

/* State pointer (pointer to function) */
extern void (*ALARM_ACTUATOR_DRIVER_state)();

#endif
```

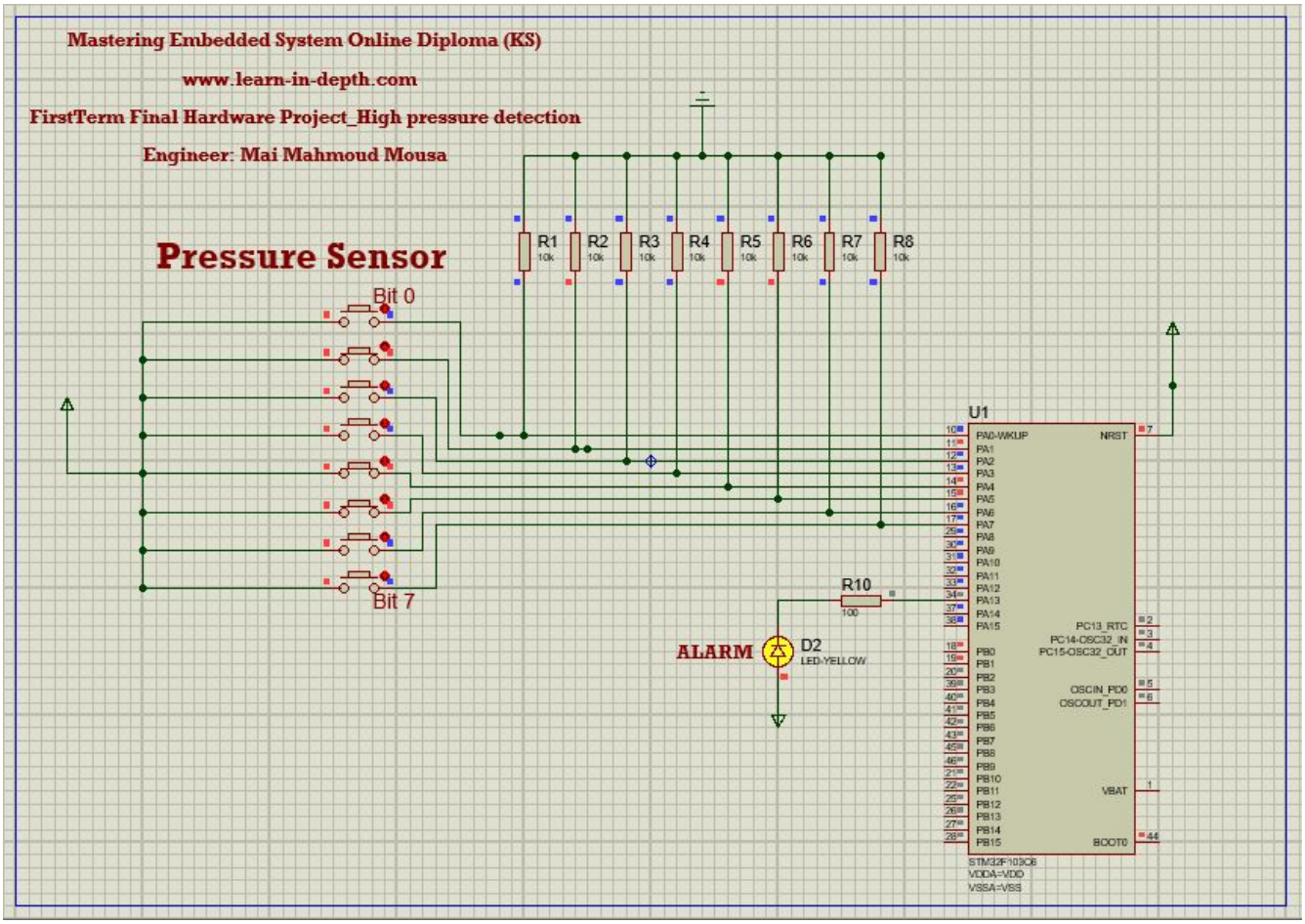
- Alarm actuator driver state machine.



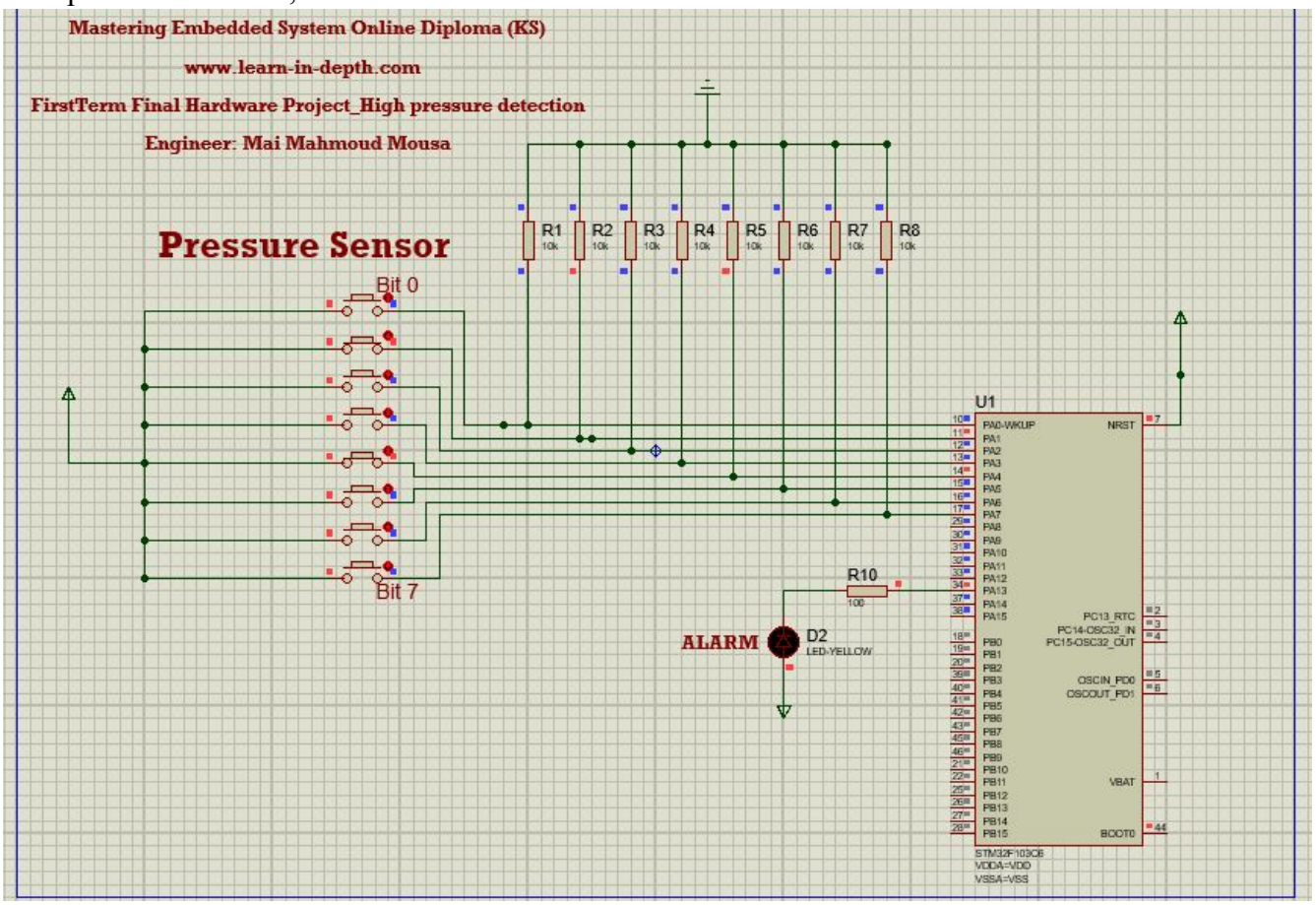
Simulation on proteus:

Here, we have two cases in simulation:

- LED is ON: pressure is greater than 20 bar.
  Now pressure is 50 bar, so LED is ON.



- LED is OFF: pressure is less than 20 bar.
  Now pressure is 18 bar, so LED is OFF.



Software analysis:

- Software building using ARM cross toolchain.

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>ls
Alarm_Actuator_Driver.c  Main_Algorithm.c  Pressure_Sensor_Driver.c  linker_script.ld
Alarm_Actuator_Driver.h  Main_Algorithm.h  Pressure_Sensor_Driver.h  main.c
Alarm_Monitor.c          Makefile          driver.c                  startup.c
Alarm_Monitor.h          Platform_Types.h  driver.h                  state.h

D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>make
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -mthumb -c  -I .  Alarm_Actuator_Driver.c -o Alarm_Actuat
or_Driver.o
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -mthumb -c  -I .  Alarm_Monitor.c -o Alarm_Monitor.o
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -mthumb -c  -I .  driver.c -o driver.o
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -mthumb -c  -I .  main.c -o main.o
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -mthumb -c  -I .  Main_Algorithm.c -o Main_Algorithm.o
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -mthumb -c  -I .  Pressure_Sensor_Driver.c -o Pressure_Se
nsor_Driver.o
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -mthumb -c  -I .  startup.c -o startup.o
arm-none-eabi-ld.exe  -T linker_script.ld Alarm_Actuator_Driver.o Alarm_Monitor.o driver.o main.o Main_Al
gorithm.o Pressure_Sensor_Driver.o startup.o   -o HighPressureDetection.elf -Map=HighPressureDetection.Ma
p
arm-none-eabi-objcopy.exe -O binary HighPressureDetection.elf HighPressureDetection.bin
Build is finished ...

D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>ls
Alarm_Actuator_Driver.c    HighPressureDetection.bin  Pressure_Sensor_Driver.c  main.c
Alarm_Actuator_Driver.h    HighPressureDetection.elf  Pressure_Sensor_Driver.h  main.o
Alarm_Actuator_Driver.o    Main_Algorithm.c           Pressure_Sensor_Driver.o  startup.c
Alarm_Monitor.c            Main_Algorithm.h           driver.c                  startup.o
Alarm_Monitor.h            Main_Algorithm.o           driver.h                  state.h
Alarm_Monitor.o            Makefile                   driver.o
HighPressureDetection.Map  Platform_Types.h           linker_script.ld
```

- Analyzing sections in output object files and final elf image.

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>ls *.o
Alarm_Actuator_Driver.o  Main_Algorithm.o          driver.o  startup.o
Alarm_Monitor.o          Pressure_Sensor_Driver.o  main.o
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-objdump.exe
 -h Pressure_Sensor_Driver.o

Pressure_Sensor_Driver.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   00000112  00000000  00000000  000000bc  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000aa  00000000  00000000  000001ce  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000084  00000000  00000000  00000278  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  000002fc  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000070  00000000  00000000  0000031c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000001f3  00000000  00000000  0000038c  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  0000057f  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00000591  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000005c  00000000  00000000  000005c4  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-objdump.exe
 -h Main_Algorithm.o

Main_Algorithm.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000078  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000004  00000000  00000000  000000ac  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  000000b0  2**2
                  ALLOC
  3 .debug_info   0000011c  00000000  00000000  000000b0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000a5  00000000  00000000  000001cc  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000064  00000000  00000000  00000271  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  000002d5  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000063  00000000  00000000  000002f5  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000001a9  00000000  00000000  00000358  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  00000501  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00000513  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000048  00000000  00000000  00000548  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-objdump.exe
 -h Alarm_Monitor.o

Alarm_Monitor.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000009c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000004  00000000  00000000  000000d0  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000d4  2**0
                  ALLOC
  3 .debug_info   0000012d  00000000  00000000  000000d4  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000aa  00000000  00000000  00000201  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000000b0  00000000  00000000  000002ab  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  0000035b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000060  00000000  00000000  0000037b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000001dd  00000000  00000000  000003db  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  000005b8  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000005ca  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000078  00000000  00000000  00000600  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-objdump.exe
 -h Alarm_Actuator_Driver.o

Alarm_Actuator_Driver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000bc  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000f0  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000f0  2**0
                  ALLOC
  3 .debug_info   00000147  00000000  00000000  000000f0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000aa  00000000  00000000  00000237  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000108  00000000  00000000  000002e1  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  000003e9  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000074  00000000  00000000  00000409  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000233  00000000  00000000  0000047d  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  000006b0  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000006c2  2**0
                  CONTENTS, READONLY
 11 .debug_frame  000000a8  00000000  00000000  000006f8  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-objdump
.exe -h HighPressureDetection.elf

HighPressureDetection.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000046c  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  0800046c  0800046c  0000846c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000024  20000000  20000000  00010000  2**2
                  ALLOC
  3 .debug_info   000007b0  00000000  00000000  00008474  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 000003fb  00000000  00000000  00008c24  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000003c0  00000000  00000000  0000901f  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000e0  00000000  00000000  000093e0  2**3
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000332  00000000  00000000  000094c0  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    00000410  00000000  00000000  000097f2  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  00009c02  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000031  00000000  00000000  00009c13  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000284  00000000  00000000  00009c44  2**2
                  CONTENTS, READONLY, DEBUGGING
```

- Analyzing symbol tables in output object files and final elf image.

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>ls *.o
Alarm_Actuator_Driver.o  Main_Algorithm.o          driver.o  startup.o
Alarm_Monitor.o          Pressure_Sensor_Driver.o  main.o
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-nm.exe
Pressure_Sensor_Driver.o
         U Delay
         U getPressureVal
00000000 T Pressure_Sensor_Driver_init
00000004 C PRESSURE_SENSOR_DRIVER_state
00000001 C PRESSURE_SENSOR_DRIVER_state_id
00000000 B pSensor_Driver_pVal
         U Set_Pressure_Val
0000000c T ST_PRESSURE_SENSOR_DRIVER_reading
0000005c T ST_PRESSURE_SENSOR_DRIVER_waiting
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-nm.exe
Main_Algorithm.o
         U High_Pressure_Detected
00000000 B Main_Algorithm_pVal
00000004 C MAIN_ALGORITHM_state
00000001 C MAIN_ALGORITHM_state_id
00000000 D pThreshold
00000000 T Set_Pressure_Val
00000030 T ST_MAIN_ALGORITHM_high_pressure_detect
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-nm.exe
Alarm_Monitor.o
00000004 C ALARM_MONITOR_state
00000001 C ALARM_MONITOR_state_id
00000000 D Alarm_Period
         U Delay
00000000 T High_Pressure_Detected
0000001c T ST_ALARM_MONITOR_alarm_off
00000034 T ST_ALARM_MONITOR_alarm_on
00000070 T ST_ALARM_MONITOR_waiting
         U Start_Alarm
         U Stop_Alarm
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-nm.exe
Alarm_Actuator_Driver.o
00000000 T Alarm_Actuator_Driver_init
00000004 C ALARM_ACTUATOR_DRIVER_state
00000001 C ALARM_ACTUATOR_DRIVER_state_id
         U Set_Alarm_actuator
00000044 T ST_ALARM_ACTUATOR_DRIVER_alarm_off
00000074 T ST_ALARM_ACTUATOR_DRIVER_alarm_on
000000a4 T ST_ALARM_ACTUATOR_DRIVER_waiting
0000000c T Start_Alarm
00000028 T Stop_Alarm
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-nm.exe
HighPressureDetection.elf
08000464 t _reset
08000068 T Alarm_Actuator_Driver_init
2000000c B ALARM_ACTUATOR_DRIVER_state
20000008 B ALARM_ACTUATOR_DRIVER_state_id
20000014 B ALARM_MONITOR_state
20000010 B ALARM_MONITOR_state_id
0800046c D Alarm_Period
080001c0 T Delay
080001e4 T getPressureVal
0800024c T GPIO_INITIALIZATION
08000124 T High_Pressure_Detected
08000324 T main
20000000 B Main_Algorithm_pVal
2000001c B MAIN_ALGORITHM_state
20000018 B MAIN_ALGORITHM_state_id
080003dc T Pressure_Sensor_Driver_init
20000020 B PRESSURE_SENSOR_DRIVER_state
20000019 B PRESSURE_SENSOR_DRIVER_state_id
20000004 B pSensor_Driver_pVal
08000470 D pThreshold
080001fc T Set_Alarm_actuator
08000364 T Set_Pressure_Val
080002cc T setup
080000ac T ST_ALARM_ACTUATOR_DRIVER_alarm_off
080000dc T ST_ALARM_ACTUATOR_DRIVER_alarm_on
0800010c T ST_ALARM_ACTUATOR_DRIVER_waiting
08000140 T ST_ALARM_MONITOR_alarm_off
08000158 T ST_ALARM_MONITOR_alarm_on
08000194 T ST_ALARM_MONITOR_waiting
08000394 T ST_MAIN_ALGORITHM_high_pressure_detect
080003e8 T ST_PRESSURE_SENSOR_DRIVER_reading
08000438 T ST_PRESSURE_SENSOR_DRIVER_waiting
08000074 T Start_Alarm
08000090 T Stop_Alarm
0800046a t Vector_handler
```

- Analyzing the final elf image:

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-readelf.exe -S
 HighPressureDetection.elf
There are 16 section headers, starting at offset 0x9f68:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .text             PROGBITS        08000000 008000 00046c 00  AX  0   0  4
  [ 2] .data             PROGBITS        0800046c 00846c 000008 00  WA  0   0  4
  [ 3] .bss              NOBITS          20000000 010000 000024 00  WA  0   0  4
  [ 4] .debug_info       PROGBITS        00000000 008474 0007b0 00      0   0  1
  [ 5] .debug_abbrev     PROGBITS        00000000 008c24 0003fb 00      0   0  1
  [ 6] .debug_loc        PROGBITS        00000000 00901f 0003c0 00      0   0  1
  [ 7] .debug_aranges    PROGBITS        00000000 0093e0 0000e0 00      0   0  8
  [ 8] .debug_line       PROGBITS        00000000 0094c0 000332 00      0   0  1
  [ 9] .debug_str        PROGBITS        00000000 0097f2 000410 01  MS  0   0  1
  [10] .comment          PROGBITS        00000000 009c02 000011 01  MS  0   0  1
  [11] .ARM.attributes   ARM_ATTRIBUTES  00000000 009c13 000031 00      0   0  1
  [12] .debug_frame      PROGBITS        00000000 009c44 000284 00      0   0  4
  [13] .shstrtab         STRTAB          00000000 009ec8 00009d 00      0   0  1
  [14] .symtab           SYMTAB          00000000 00a1e8 0004b0 10     15  42  4
  [15] .strtab           STRTAB          00000000 00a698 000375 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), T (TLS), E (exclude), x (unknown)
  O (extra OS processing required) o (OS specific), p (processor specific)
```

```
D:\KS\6.First Term End\First Term Project_HW\Implementation\Code\LastVersion>arm-none-eabi-readelf.exe -a
 HighPressureDetection.elf
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x8000000
  Start of program headers:          52 (bytes into file)
  Start of section headers:          73256 (bytes into file)
  Flags:                             0x5000002, has entry point, Version5 EABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         2
  Size of section headers:           40 (bytes)
  Number of section headers:         16
  Section header string table index: 13
```