Mastering Embedded System Online Diploma

https://www.learn-in-depth.com/

First Term-Final Project2

Abdullah Mohamed Kortam

My Profile

https://www.learn-in-depth.com/online-diploma/abdullahkortam60%40gmail.com

High Pressure Detection

System design sequence:

1) Case study:

* The client expects the software to detect the high pressure above 20 bars and inform the crew in the cabin.
* If the pressure is above the threshold, turn on the LED alarm and its duration is 60 sec.

Assumption:

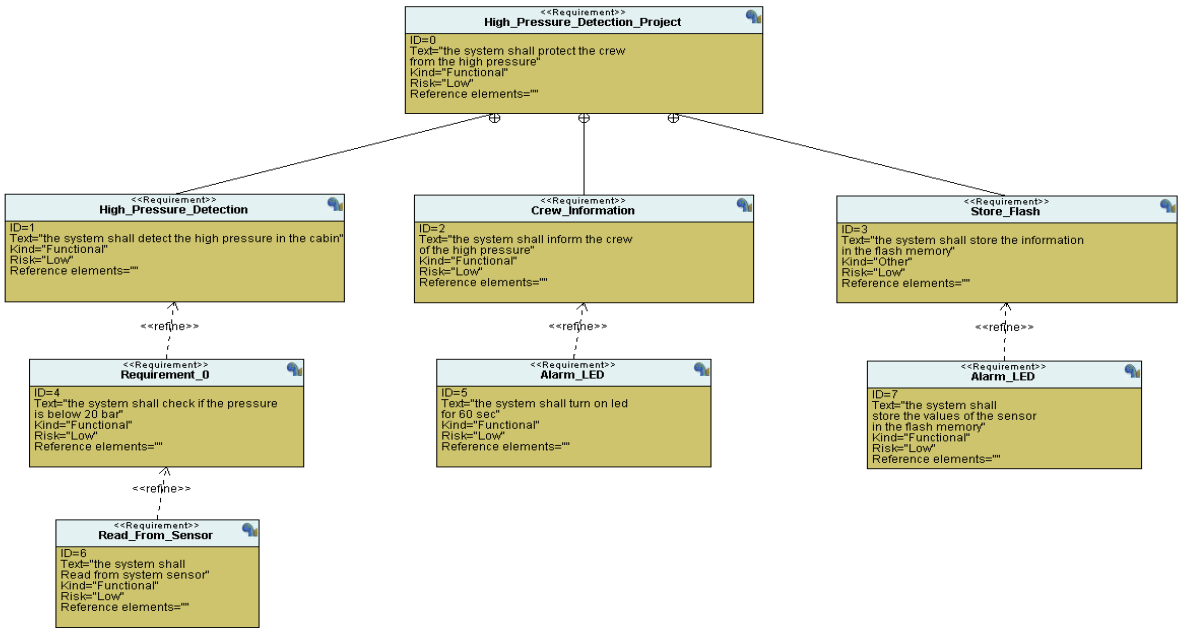
The controller setup and shut down procedures are not modeled.
The controller's maintenance is not modeled.
The pressure sensor never fails.
The alarm never fails.
The controller never faces a power cut.

2) Implementation Method:

The V-Model method is chosen to implementing this system.

3) The system Requirements:

The required UML diagram

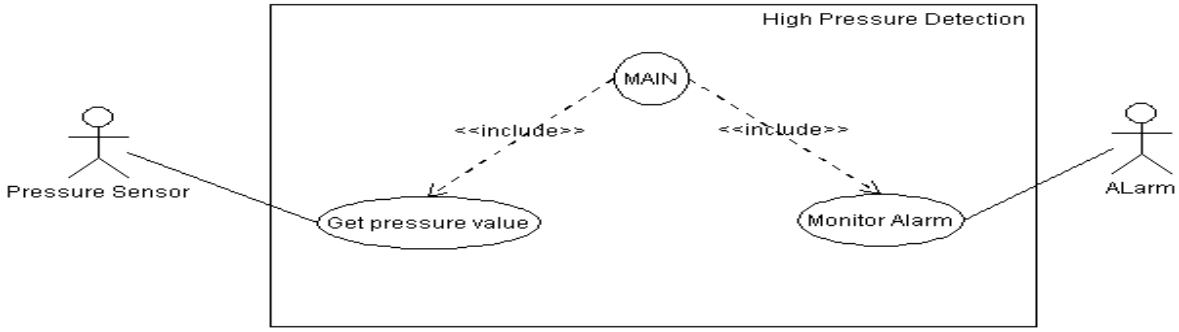


4) Hardware:

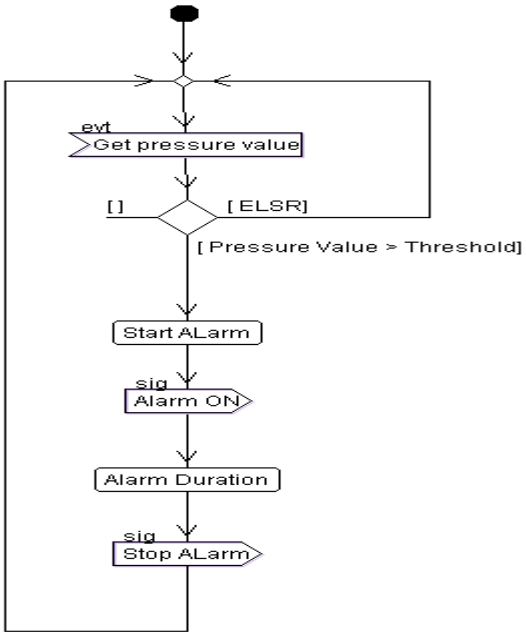
µ controller : STM32F103C6.
Sensor: Pressure Sensor
Alarm: LED.

5) System Analysis:

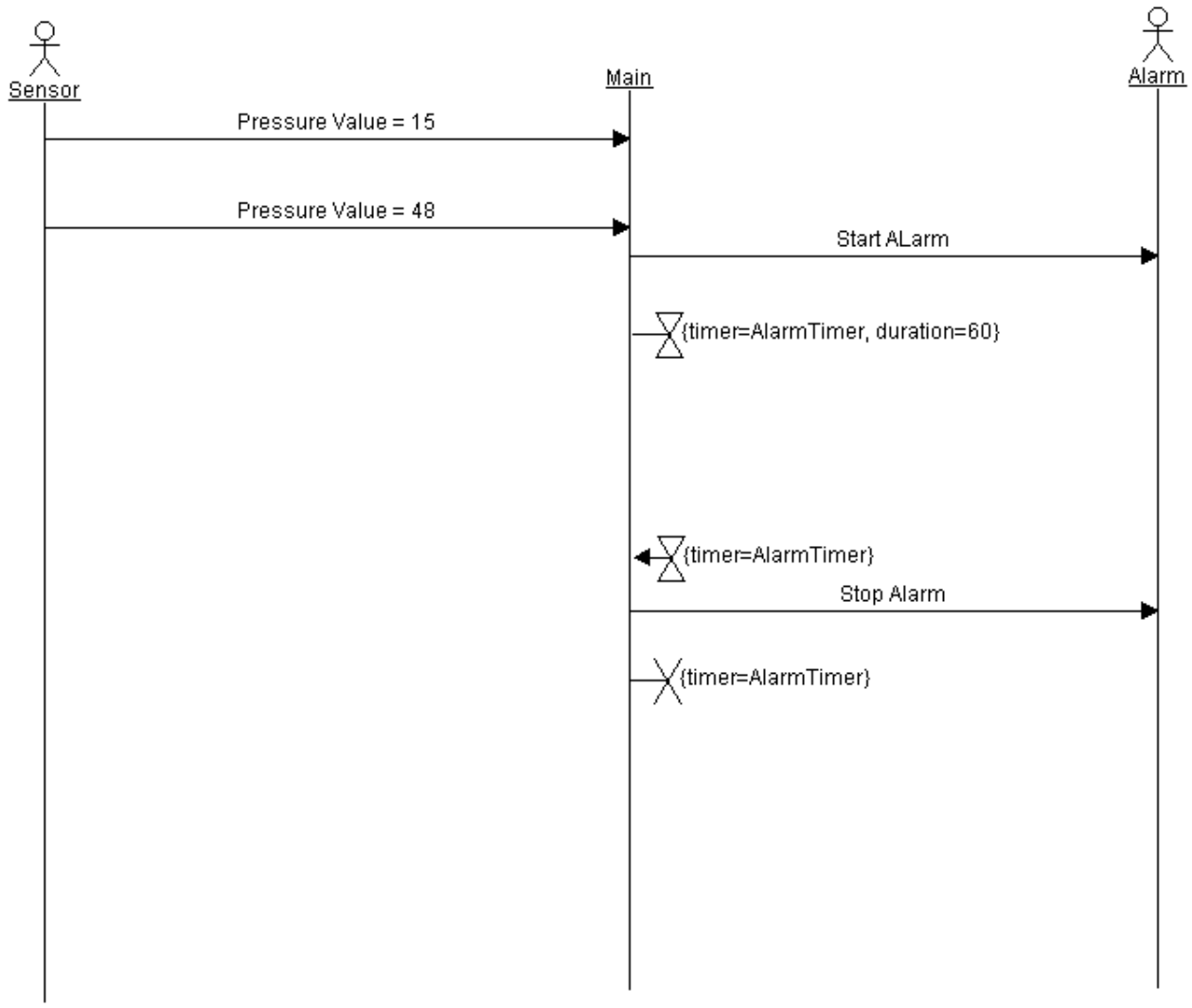
1) Use Case Diagram:



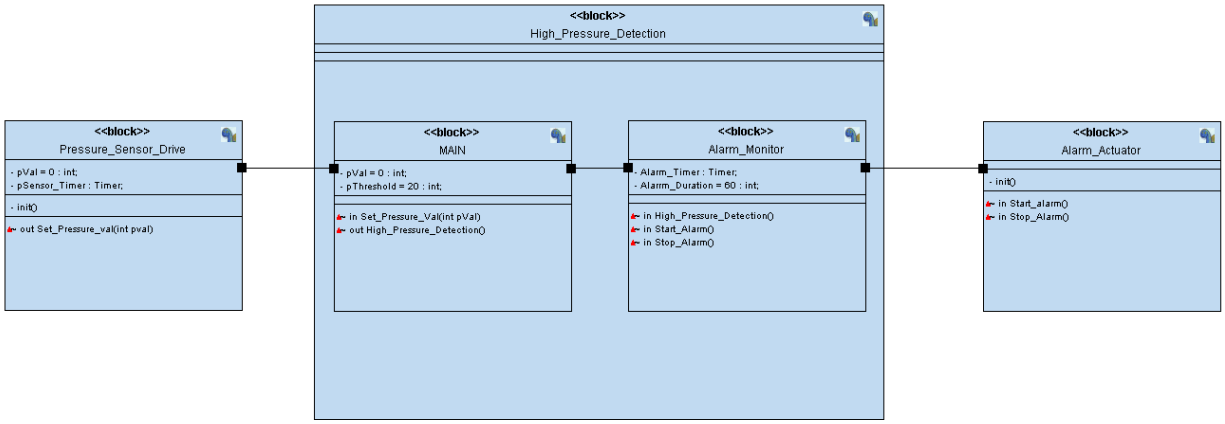
2) Activity Diagram:



2) Sequence Diagram:



6) System Analysis:



7) Software:

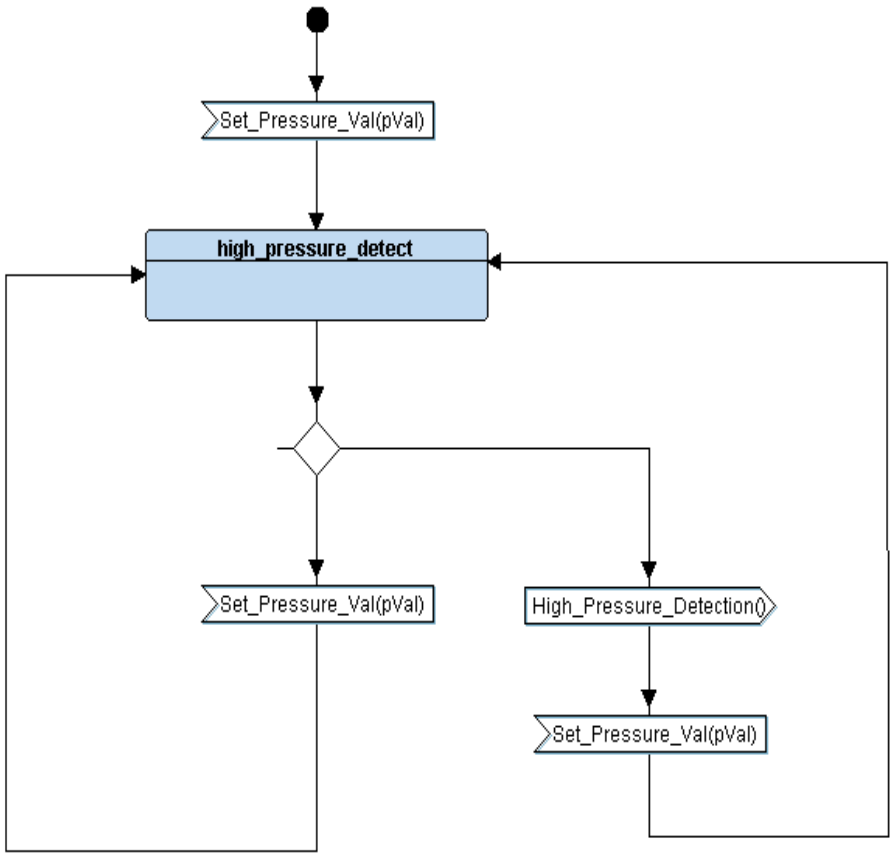
Main Algorithm.h

```
/*****************************************************************************************************************
 *  Main_Algorithm.h
 *  Second_Term_project HW
 *  Created on: ١٢/٠٩/٢٠٢٢
 *  Author: Abdullah Kortam
 *****************************************************************************************************************/

#ifndef MAIN_ALGORITHM_H_
#define MAIN_ALGORITHM_H_
#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_H_ */
```

Main Algorithm.c

```c
/*****************************************************************************************************************
 *  Main_Algorithm.c
 *  Second_Term_project HW
 *  Created on: ١٢<0x200f>/٠٩<0x200f>/٢٠٢٢
 *  Author: Abdullah Kortam
 *****************************************************************************************************************/
#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(); }
}
```

State machine:



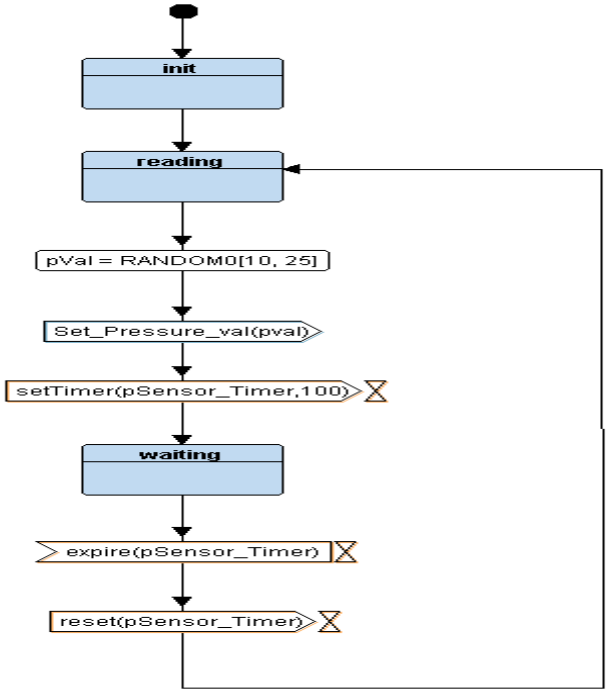
Pressure Driver.h:

```c
/*****************************************************************************************************************
 *  Pressure_Sensor_Driver.h
 *  Second_Term_project HW
 *  Created on: ١٢<0x200f>/٠٩<0x200f>/٢٠٢٢
 *  Author: Abdullah Kortam
 *****************************************************************************************************************/

#ifndef PRESSURE_SENSOR_DRIVER_H_
#define PRESSURE_SENSOR_DRIVER_H_
#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_H_ */
```

Pressure Driver.c

```c
/*****************************************************************************************
 *  Pressure_Sensor_Driver.c
 *  Second_Term_project HW
 *  Created on: ١٢<0x200f>/٠٩<0x200f>/٢٠٢٢
 *  Author: Abdullah Kortam
 *****************************************************************************************/
#include "Pressure_Sensor_Driver.h"
#include "state.h"
#include "driver.h"

/* 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);
}
```

State machine



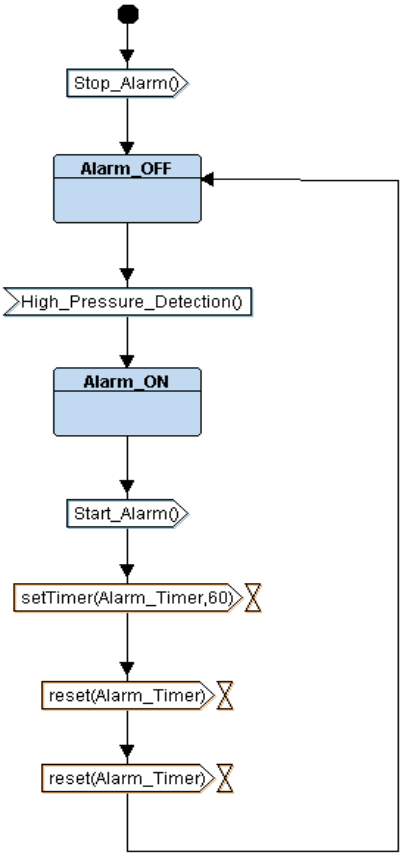
Alarm monitor.h

```
/*****************************************************************************
 *  Alarm_Monitor.h
 *  Second_Term_project HW
 *  Created on: ١٢<0x200f>/٠٩<0x200f>/٢٠٢٢
 *  Author: Abdullah Kortam
 *****************************************************************************/

#ifndef ALARM_MONITOR_H_
#define ALARM_MONITOR_H_
#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_H_ */
```

Alarm monitor.c

```c
/*****************************************************************************************
 *  Alarm_Monitor.c
 *  Second_Term_project HW
 *  Created on: ١٢<0x200f>/٠٩<0x200f>/٢٠٢٢
 *  Author: Abdullah Kortam
 *****************************************************************************************/
#include "Alarm_Monitor.h"
#include "state.h"
#include "driver.h"

/* Variables */
int Alarm_Period = 60;

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

/* Input signal(s) implementation */
void High_Pressure_Detected (void)
{
    /* Set the next state */
    ALARM_MONITOR_state = STATE(ALARM_MONITOR_alarm_on);
}

/* State(s) implementation */
STATE_define(ALARM_MONITOR_alarm_off)
{
    /* State name */
    ALARM_MONITOR_state_id = ALARM_MONITOR_alarm_off;
    /* 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);
}
```

State machine



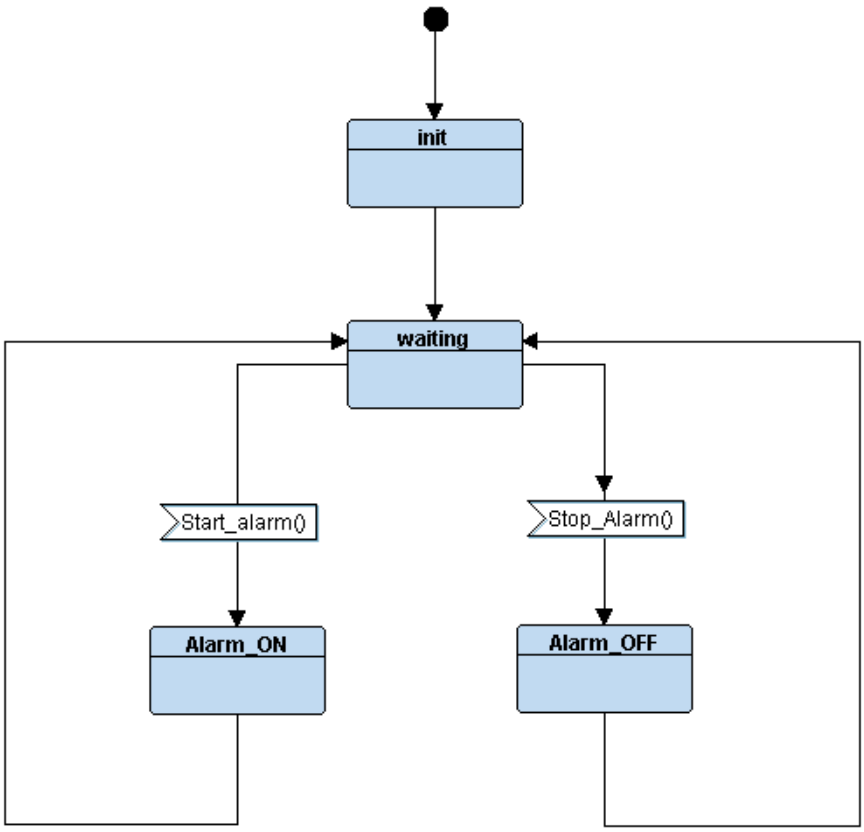
Alarm actuator.h:

```c
/*****************************************************************************
 *  Alarm_Actuator_Driver.h
 *  Second_Term_project HW
 *  Created on: ١٢<0x200f>/٠٩<0x200f>/٢٠٢٢
 *  Author: Abdullah Kortam
 *****************************************************************************/
#ifndef ALARM_ACTUATOR_DRIVER_H_
#define ALARM_ACTUATOR_DRIVER_H_
#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_H_ */
```

Alarm actuator.c

```c
/* 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)
{
    /* Set the next state */
    ALARM_ACTUATOR_DRIVER_state = STATE(ALARM_ACTUATOR_DRIVER_alarm_on);
}

void Stop_Alarm (void)
{
    /* Set the next state */
    ALARM_ACTUATOR_DRIVER_state = STATE(ALARM_ACTUATOR_DRIVER_alarm_off);
}

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

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

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

State machine



Software Building:

```
mkm@DESKTOP-58OJK14 MINGW64 /d/test1/HW
$ make all
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -mthumb -c  -I .  Alarm_Actuator
_Driver.c -o Alarm_Actuator_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 dr
iver.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_Senso
r_Driver.c -o Pressure_Sensor_Driver.o
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -gdwarf-2 -mthumb -c  -I .  startup.c -o s
tartup.o
arm-none-eabi-ld.exe  -T linker_script.ld Alarm_Actuator_Driver.o Alarm_Monitor.
o driver.o main.o Main_Algorithm.o Pressure_Sensor_Driver.o startup.o   -o HighP
ressureDetection.elf -Map=HighPressureDetection.Map
arm-none-eabi-objcopy.exe -O binary HighPressureDetection.elf HighPressureDetect
ion.bin
----------------------------First Term is finished----------------------------
```

Objfile image:

```
mkm@DESKTOP-580JK14 MINGW64 ~/Desktop/First_Term_Project_2
$ 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    000001c4  00000000  00000000  0000038c  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  00000550  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00000562  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000005c  00000000  00000000  00000598  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

```
mkm@DESKTOP-580JK14 MINGW64 ~/Desktop/First_Term_Project_2
$ 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    0000017a  00000000  00000000  00000358  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  000004d2  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000004e4  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000048  00000000  00000000  00000518  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

```
mkm@DESKTOP-580JK14 MINGW64 ~/Desktop/First_Term_Project_2
$ 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   00000062  00000000  00000000  0000037b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000001ae  00000000  00000000  000003dd  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  0000058b  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  0000059d  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000078  00000000  00000000  000005d0  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

```
mkm@DESKTOP-580JK14 MINGW64 ~/Desktop/First_Term_Project_2
$ 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   00000076  00000000  00000000  00000409  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000204  00000000  00000000  0000047f  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  00000683  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00000695  2**0
                  CONTENTS, READONLY
 11 .debug_frame  000000a8  00000000  00000000  000006c8  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

```
mkm@DESKTOP-58OJK14 MINGW64 ~/Desktop/First_Term_Project_2
$ 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         000004d4  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  20000000  080004d4  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001024  20000008  080004dc  00010008  2**2
                  ALLOC
  3 .debug_info   000008b3  00000000  00000000  00010008  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 000004ab  00000000  00000000  000108bb  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000424  00000000  00000000  00010d66  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000e0  00000000  00000000  0001118a  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000377  00000000  00000000  0001126a  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    0000047c  00000000  00000000  000115e1  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  00011a5d  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00011a6e  2**0
                  CONTENTS, READONLY
 11 .debug_frame  000002d0  00000000  00000000  00011aa4  2**2
                  CONTENTS, READONLY, DEBUGGING
```

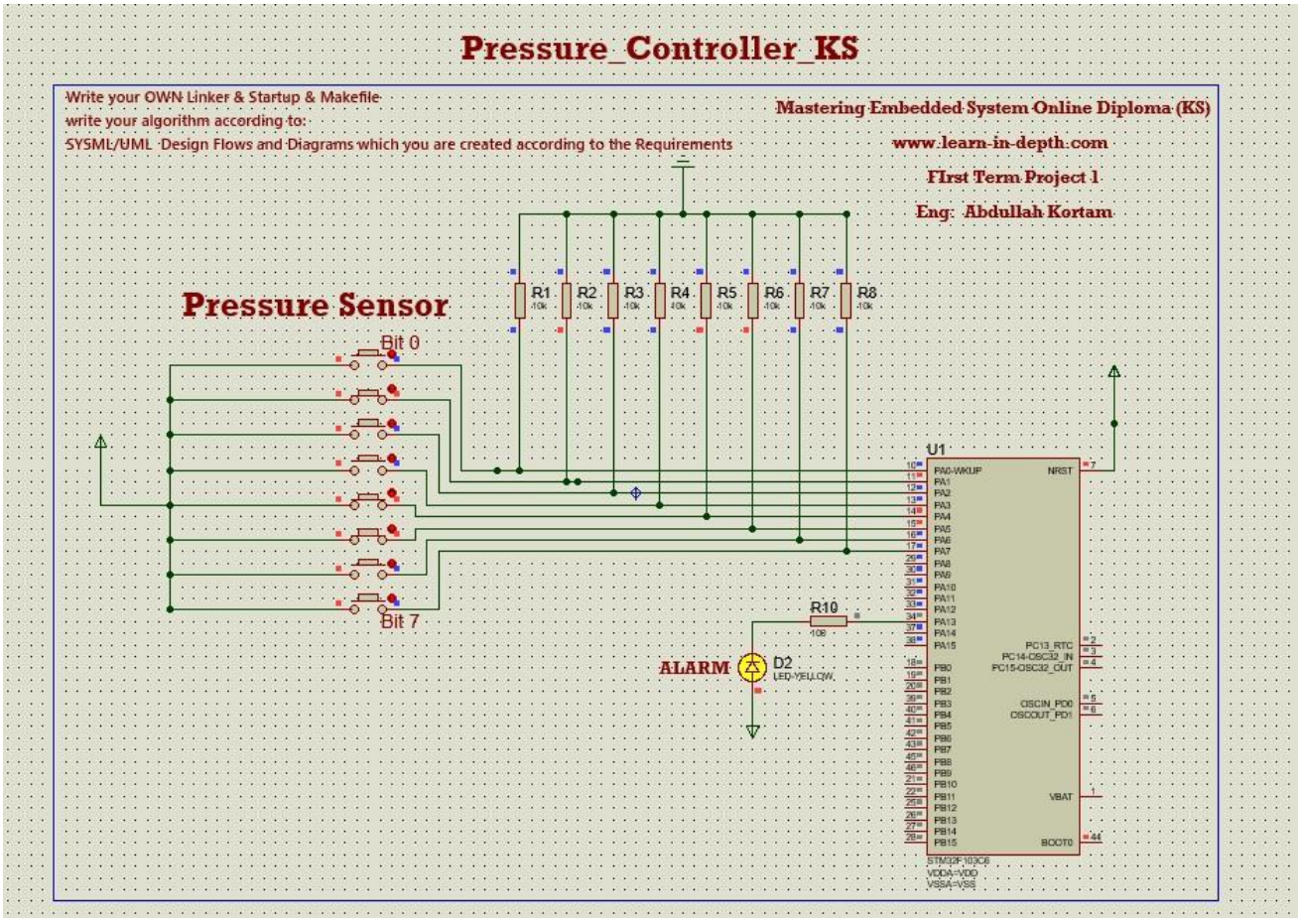
```
mkm@DESKTOP-58OJK14 MINGW64 ~/Desktop/First_Term_Project_2
$ arm-none-eabi-readelf.exe  -S HighPressureDetection.elf
There are 16 section headers, starting at offset 0x11e14:

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 0004d4 00  AX  0   0  4
  [ 2] .data             PROGBITS        20000000 010000 000008 00  WA  0   0  4
  [ 3] .bss              NOBITS          20000008 010008 001024 00  WA  0   0  4
  [ 4] .debug_info       PROGBITS        00000000 010008 0008b3 00      0   0  1
  [ 5] .debug_abbrev     PROGBITS        00000000 0108bb 0004ab 00      0   0  1
  [ 6] .debug_loc        PROGBITS        00000000 010d66 000424 00      0   0  1
  [ 7] .debug_aranges    PROGBITS        00000000 01118a 0000e0 00      0   0  1
  [ 8] .debug_line       PROGBITS        00000000 01126a 000377 00      0   0  1
  [ 9] .debug_str        PROGBITS        00000000 0115e1 00047c 01  MS  0   0  1
  [10] .comment          PROGBITS        00000000 011a5d 000011 01  MS  0   0  1
  [11] .ARM.attributes   ARM_ATTRIBUTES  00000000 011a6e 000033 00      0   0  1
  [12] .debug_frame      PROGBITS        00000000 011aa4 0002d0 00      0   0  4
  [13] .shstrtab         STRTAB          00000000 011d74 00009d 00      0   0  1
  [14] .symtab           SYMTAB          00000000 012094 000570 10     15  40  4
  [15] .strtab           STRTAB          00000000 012604 000409 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)
```

```
mkm@DESKTOP-58OJK14 MINGW64 ~/Desktop/First_Term_Project_2
$ arm-none-eabi-nm.exe  HighPressureDetection.elf
20000010 B _E_bss
20000008 D _E_data
080004d4 T _E_text
20000008 B _S_bss
20000000 D _S_data
20001010 B _stack_top
0800001c T Alarm_Actuator_Driver_init
20001014 B ALARM_ACTUATOR_DRIVER_state
20001010 B ALARM_ACTUATOR_DRIVER_state_id
2000101c B ALARM_MONITOR_state
20001018 B ALARM_MONITOR_state_id
20000000 D Alarm_Period
08000418 W Bus_Fault_Handler
08000418 T Default_Handler
08000174 T Delay
08000198 T getPressureVal
08000200 T GPIO_INITIALIZATION
08000418 W H_Fault_Handler
080000d8 T High_Pressure_Detected
080002d8 T main
20000008 B Main_Algorithm_pVal
20001024 B MAIN_ALGORITHM_state
20001020 B MAIN_ALGORITHM_state_id
08000418 W MM_Fault_Handler
08000418 W NMI_Handler
08000390 T Pressure_Sensor_Driver_init
20001028 B PRESSURE_SENSOR_DRIVER_state
20001021 B PRESSURE_SENSOR_DRIVER_state_id
2000000c B pSensor_Driver_pVal
20000004 D pThreshold
08000424 T Reset_Handler
080001b0 T Set_Alarm_actuator
08000318 T Set_Pressure_Val
08000280 T setup
08000060 T ST_ALARM_ACTUATOR_DRIVER_alarm_off
08000090 T ST_ALARM_ACTUATOR_DRIVER_alarm_on
080000c0 T ST_ALARM_ACTUATOR_DRIVER_waiting
080000f4 T ST_ALARM_MONITOR_alarm_off
0800010c T ST_ALARM_MONITOR_alarm_on
08000148 T ST_ALARM_MONITOR_waiting
08000348 T ST_MAIN_ALGORITHM_high_pressure_detect
0800039c T ST_PRESSURE_SENSOR_DRIVER_reading
080003ec T ST_PRESSURE_SENSOR_DRIVER_waiting
08000028 T Start_Alarm
08000044 T Stop_Alarm
08000418 W Usage_Fault_Handler
08000000 T vectors
```

Simulation:

Case (1): pressure above 20 bar

Pressure = 48 bar → LED ON For 60 sec.



Case (2): pressure below 20 bar.

Pressure = 18 bar → LED OFF.