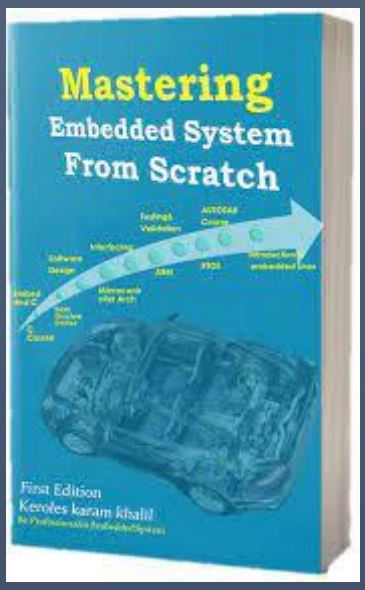


Pressure Detection Project

Mastering Embedded System Online Diploma

www.learn-in-depth.com

First Term Project (Final Project 1)

Eng. Ahmed Nabil Mahmoud

https://www.learn-in-depth.com/online-diploma/ahmed.nabil.9711%40gmail.com

Contents

1. case study

- Requirements

1. Pressure detector that informs the cabin crew when pressure exceeds a threshold of 20 Bar.

2. The system informing is acting by a Led-alarm which should last for 60 sec.
3. The system should keep tracking for pressure values.

- Assumptions

1. No setup or shut down for micro-controller.

2. No maintenance for micro-controller.

3. Pressure sensor provides accurate readings.

4. Both sensor & actuator never fail.

5. No power constrains

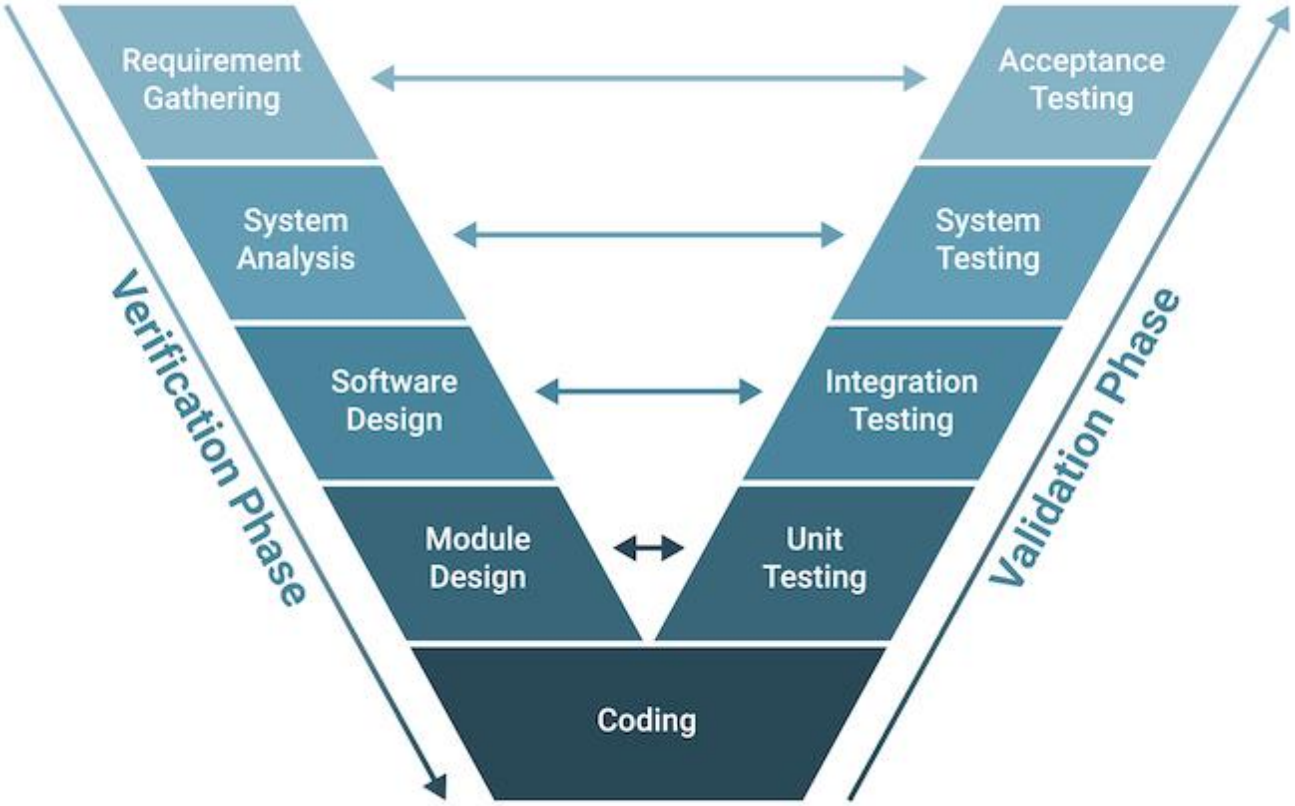
- Versioning

The probability of adding a feature to store pressure values versus time in a flash memory in the next version.

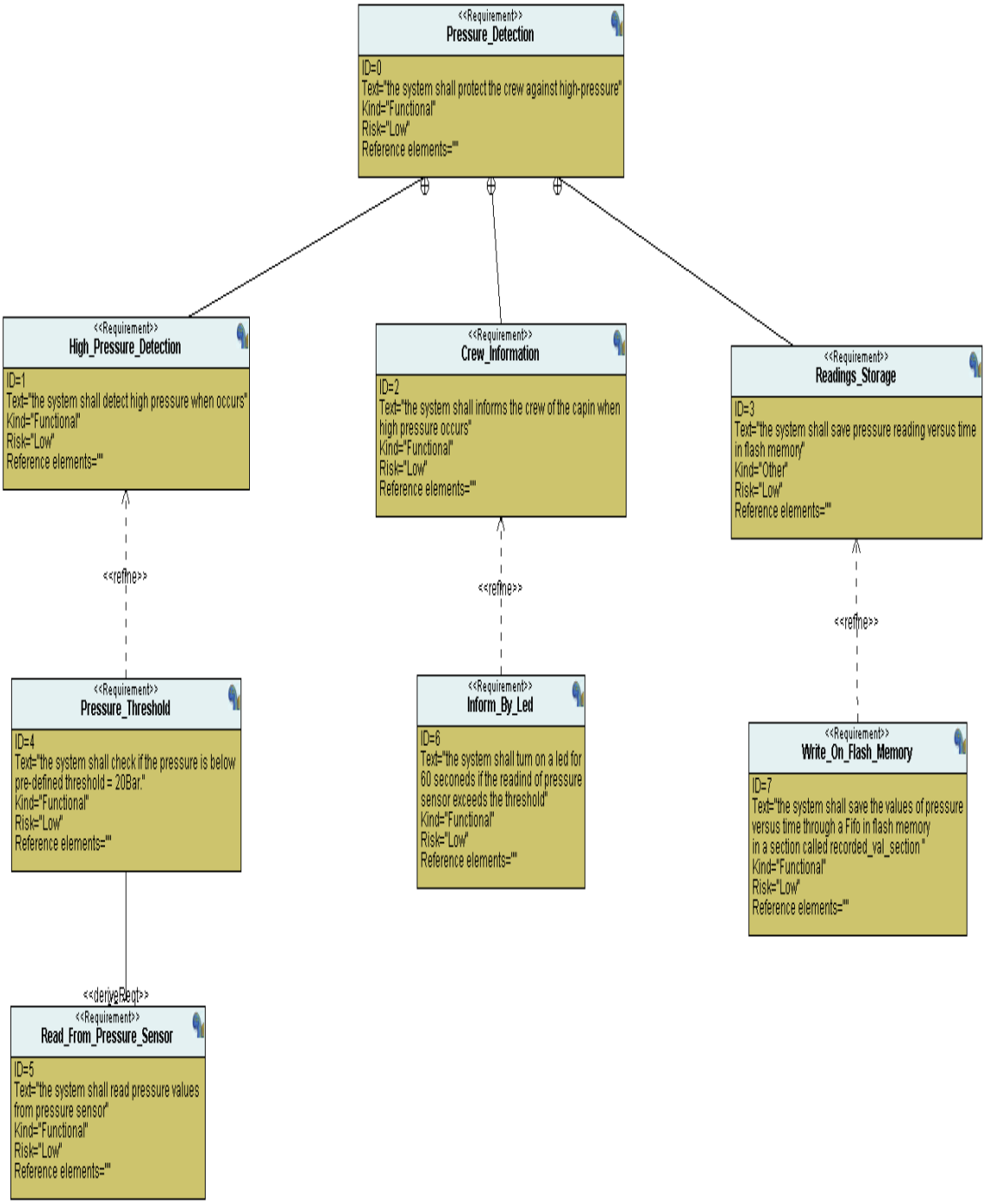
2. Method

- Software developing life cycle & software testing life cycle

The (SDLC) & (STLC) will be approached according to the V-Model.



3. Requirement Diagram



4. Space Exploration

Micro-controller : **stm32f103c8t6** SoC as it meets all technical requirments needed for this project as it is marked by : small size and contains acceptable flash memory as well as being cost efficient and have a suitable processor which is : **ARM Cortex-M3** 32bit with 72 MHz operating frequency.

- Features

operating voltage range -> 2 : 3.6 V.

64Kbytes Flash memory.

20KbytesSRam.

CRC calaculation unit , 96bit unique id.

Two 12bit, 1µs A/D converter (up to 10 channels).

7 channel DMA controller , 3 genral purpose timer & 1 advanced

Controller timer.

37 fast I/O ports.

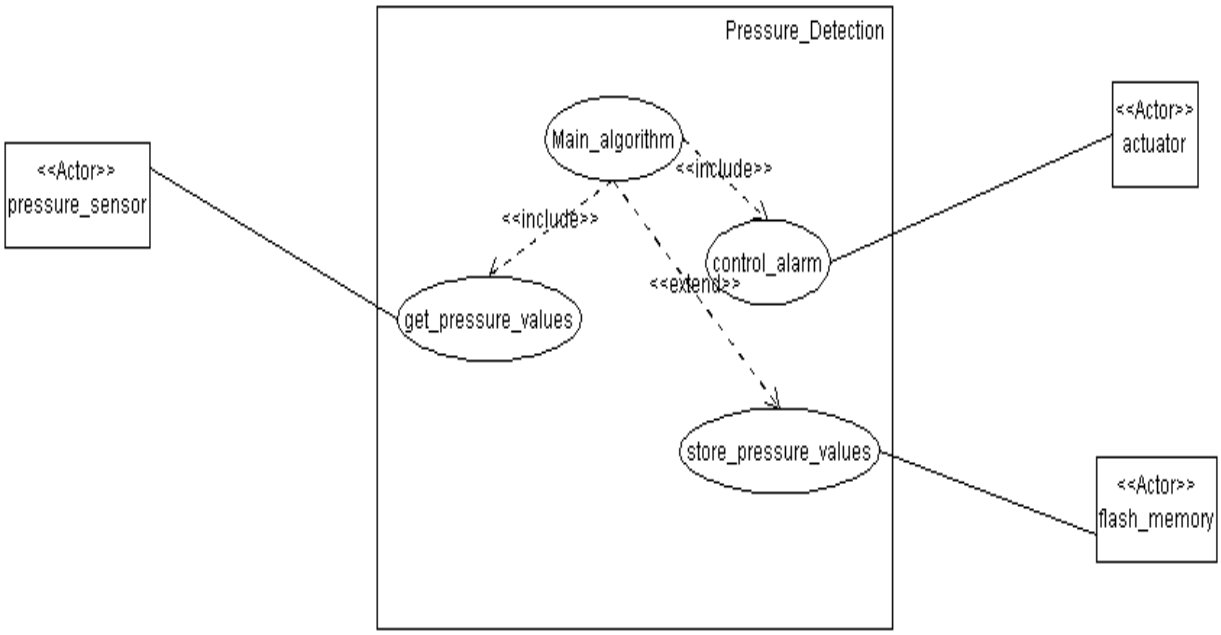
Serial wire debug (SWD) & JTAG Interfaces .

Two SPI , two I2C , three USART , one USB & one CAN interfaces.

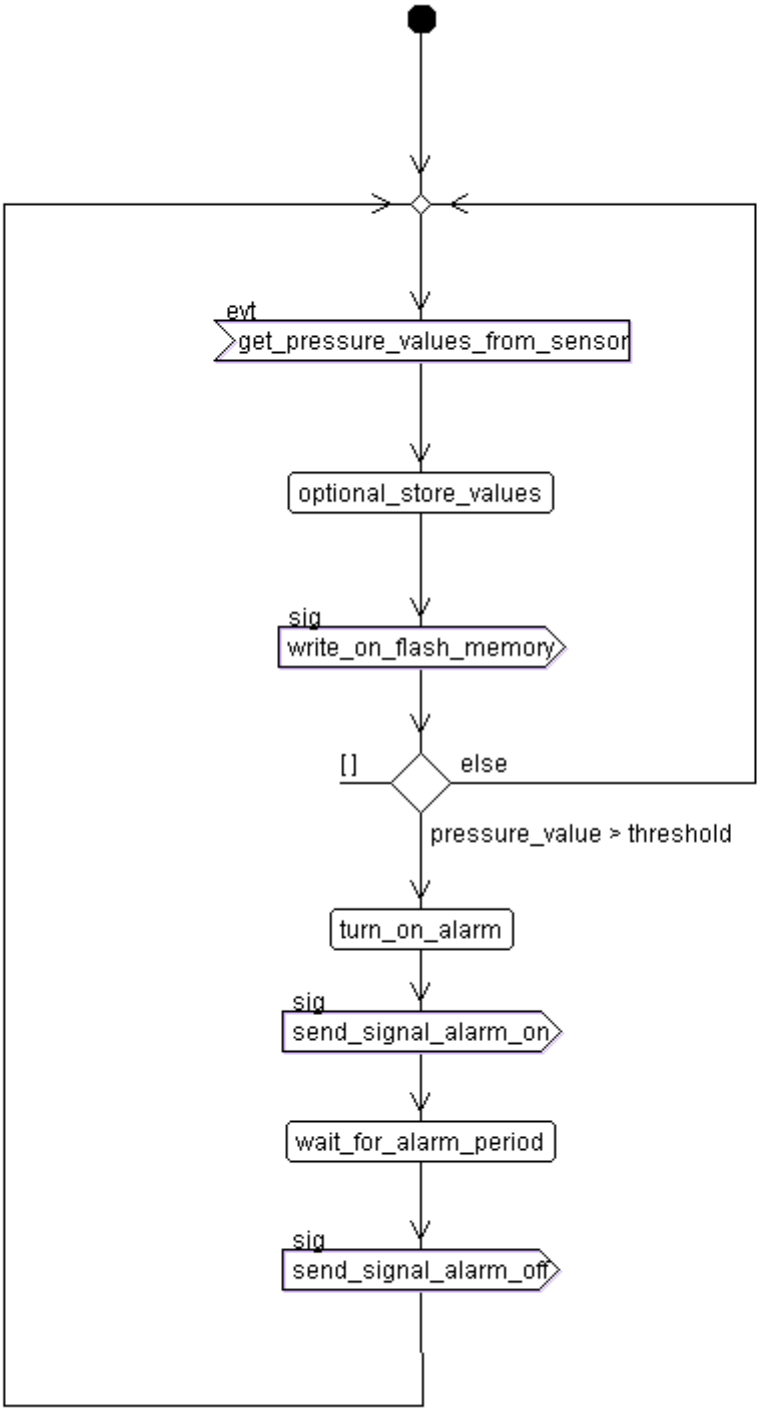
Ambient operating temperature range from -40˚C to 85˚C.

5. System Analysis

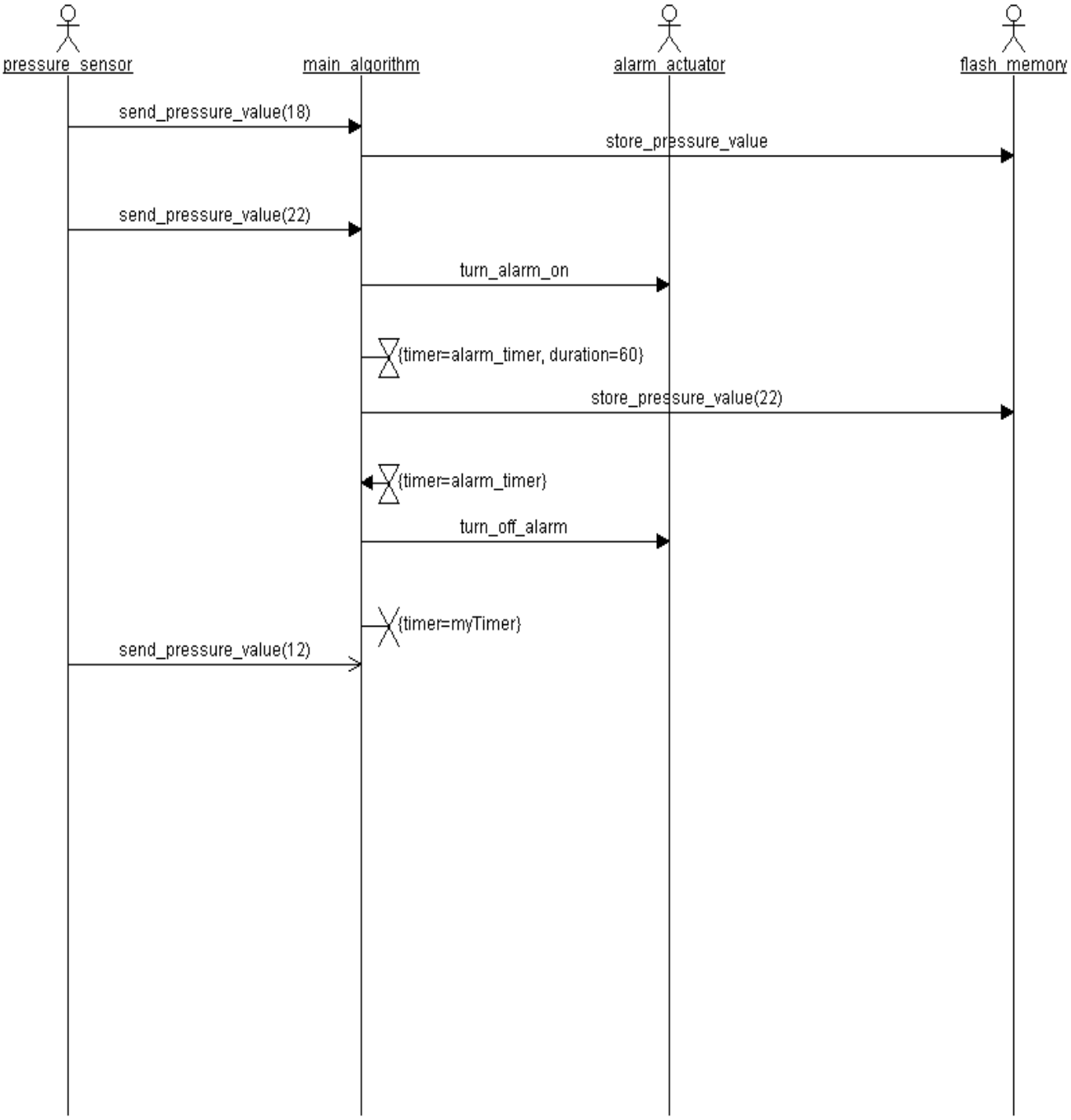
- Use Case Diagram



- Activity Diagram

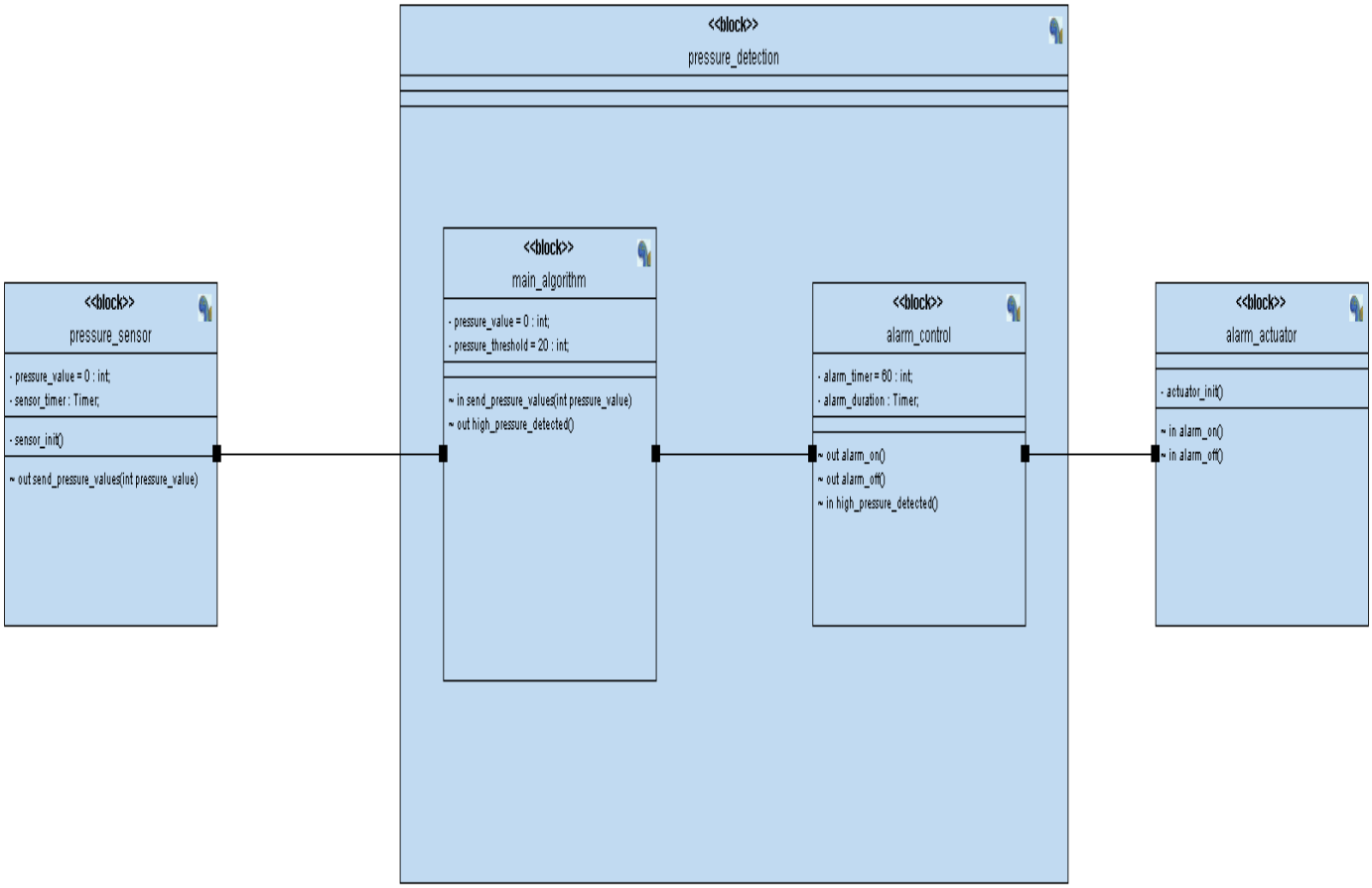


- Sequence Diagram

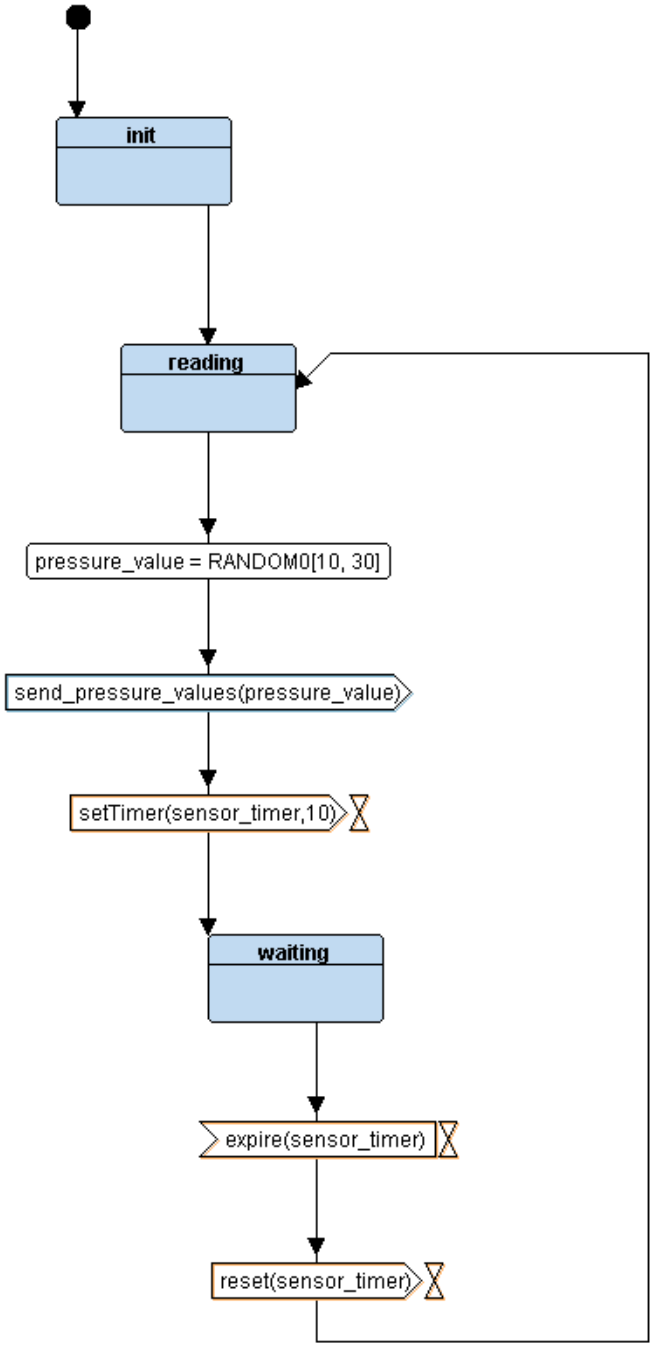


6. System Design

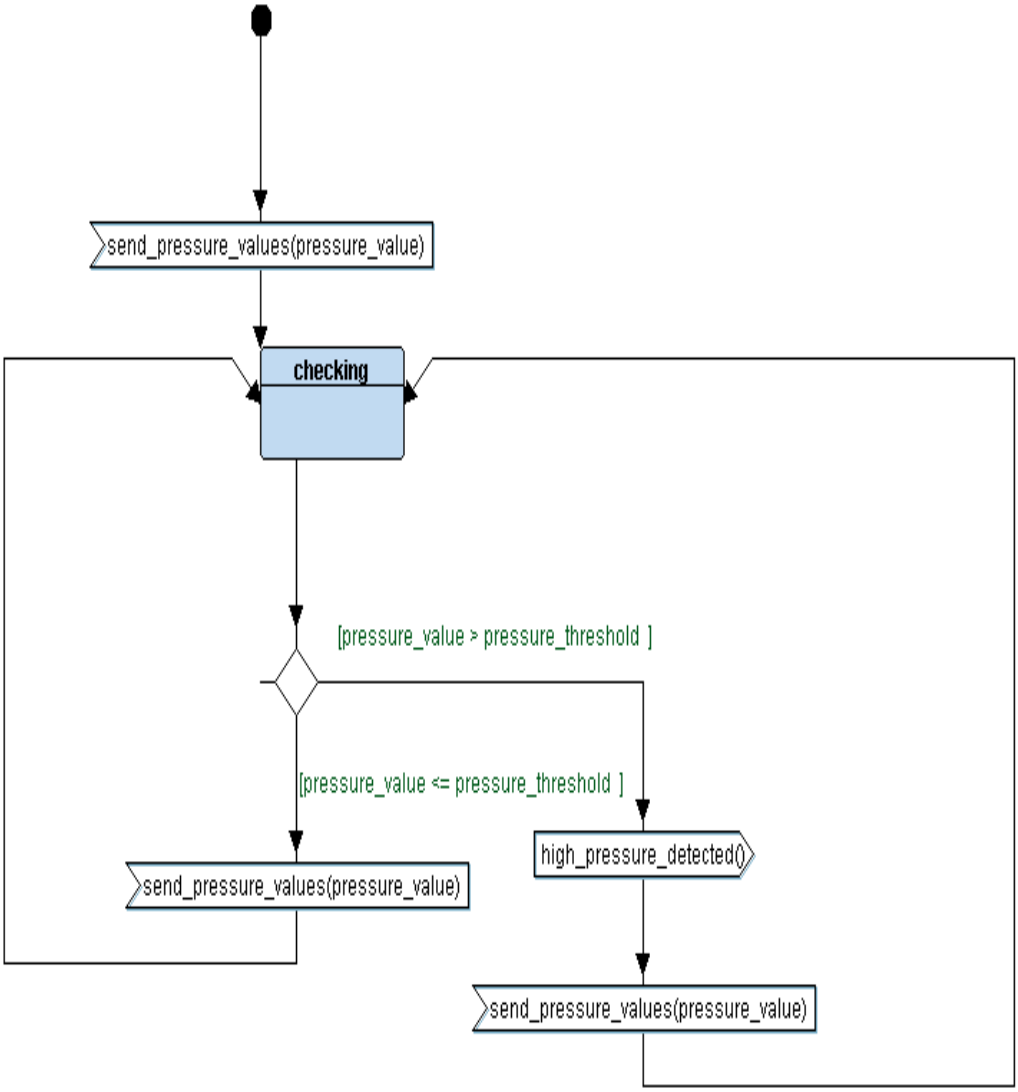
- Block Diagram



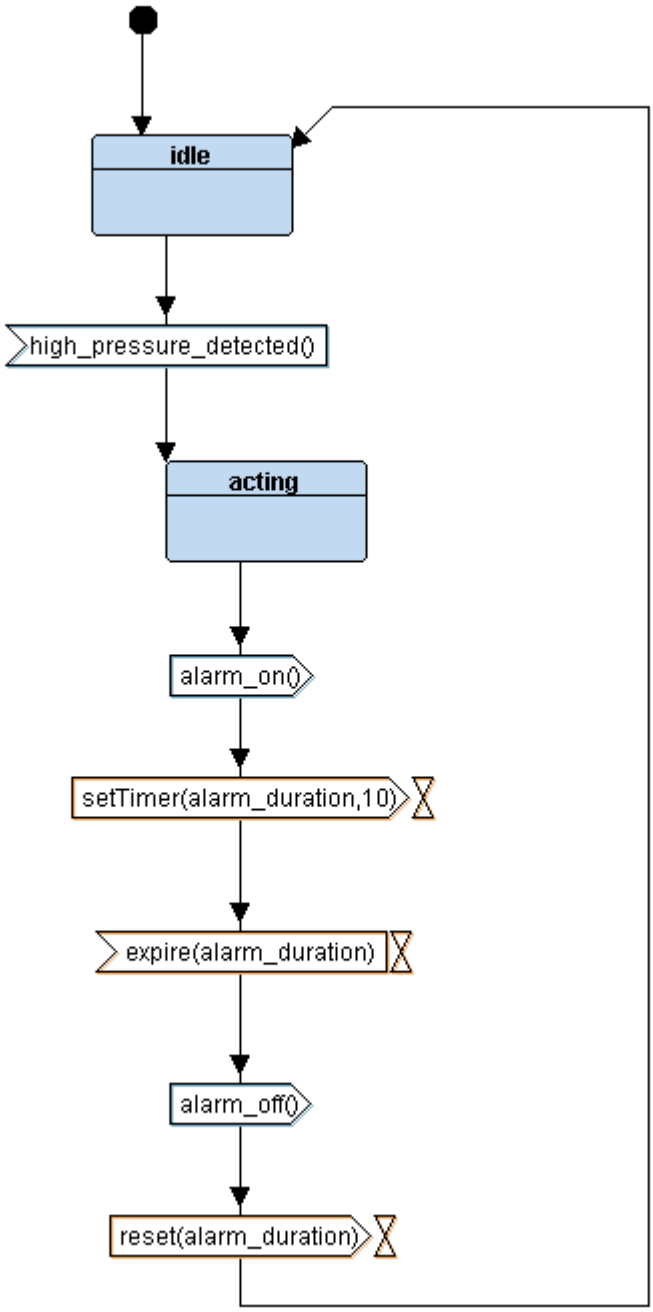
- State Machine For Pressure_Sensor



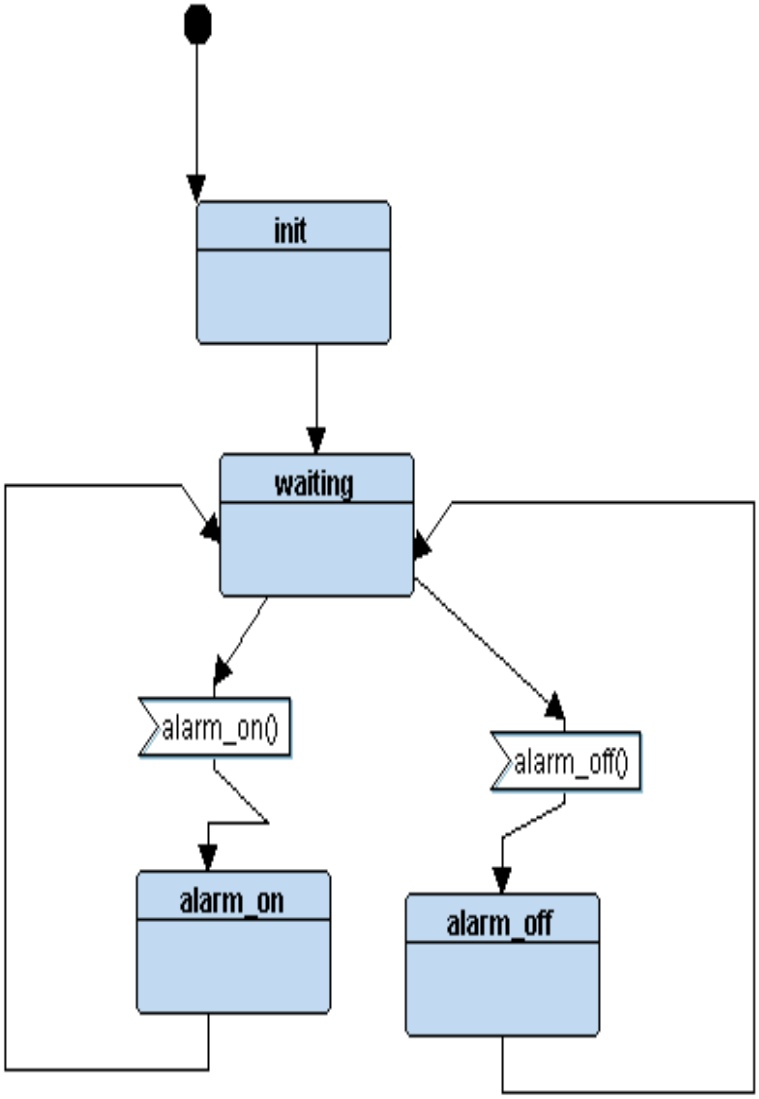
- State Machine For Main Algorithm



- State Machine For Alarm_Control



- State Machine For Alarm_Actuator



7. C codes

- Pressure_sensor_.c/.h

.h file

```c
/*
 * pressure_sensor.h
 *
 *  Created on: Oct 15, 2023
 *      Author: ahmed
 */

#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_

#include "States.h"

 enum
{
    reading_t,
    waiting_t
    }sensor_state_id;

void sensor_init();
state_define(reading);
state_define(waiting);

#endif /* PRESSURE_SENSOR_H_ */

```

.c file

untitled

```c
/*
 * pressure_sensor.c
 *
 *  Created on: Oct 15, 2023
 *      Author: ahmed
 */


#include "pressure_sensor.h"
#include "driver.h"

void (*sensor_state_ptr)();    //ptr to function of sensor state
int Pressure_value;
int sensor_pull_time = 5000000;  //by some calculations this number is equivalent to 10 sec. delay
void sensor_init()
{
    sensor_state_id = reading_t;
    sensor_state_ptr = state(reading);
    Pressure_value = 0;
}

state_define(reading)
{
    Pressure_value = getPressureVal();
    send_pressure_value(Pressure_value);                    //state action
    sensor_state_ptr = state(waiting);
}

state_define(waiting)
{
    sensor_state_id = waiting_t;
    Delay(sensor_pull_time);
    sensor_state_ptr = state(reading);
      state(reading)();

}
```

- Main_algorithm_.c/.h

.h file

```c
/*
 * main_algorithm.h
 *
 *  Created on: Oct 15, 2023
 *      Author: ahmed
 */

#ifndef MAIN_ALGORITHM_H_
#define MAIN_ALGORITHM_H_

#include "States.h"
state_define(checking);
 enum
{
    checking_t
    }main_state_id;
#endif /* MAIN_ALGORITHM_H_ */

```

.c file

```c
/*
 * main_algorithm.c
 *
 *  Created on: Oct 15, 2023
 *      Author: ahmed
 */

#include "main_algorithm.h"
#define pressure_threshold 20

int pressure_value_m;
void (* main_state_ptr)() = state(checking);
state_define(checking)
{
    main_state_id = checking_t;
    if ( pressure_value_m > pressure_threshold)
    {
        high_pressure_detected();                    //state action
    }
}

void send_pressure_value(int value)
{
    pressure_value_m= value;

    }

```

- Alarm_control_.c/.h

.h file

```c
/*
 * alarm_control.h
 *
 *  Created on: Oct 15, 2023
 *      Author: ahmed
 */

#ifndef ALARM_CONTROL_H_
#define ALARM_CONTROL_H_

#include "States.h"

 enum
{
    idle_t,
    acting_t
    }alarm_control_id;

    state_define(idle);
    state_define(acting);

#endif /* ALARM_CONTROL_H_ */

```

.c file

```c
/*
 * alarm_control.c
 *
 *  Created on: Oct 15, 2023
 *      Author: ahmed
 */


#include "alarm_control.h"
void (*alarm_control_ptr)() = state(idle);

state_define(idle)
{
    alarm_control_id = idle_t;
    set_alarm_off();
    }
state_define(acting)
{
    alarm_control_id = acting_t;
    set_alarm_on();                                  //state action
    alarm_control_ptr = state(idle);
    }

void high_pressure_detected()
{
   alarm_control_ptr = state(acting);

    }

```

- Alarm_actuator_.c/.h

.h file

```c
/*
 * alarm_actuator.h
 *
 *  Created on: Oct 15, 2023
 *      Author: ahmed
 */

#ifndef ALARM_ACTUATOR_H_
#define ALARM_ACTUATOR_H_

#include "States.h"
#include "driver.h"

 enum
{
    alarm_waiting_t,
    alarm_on_t,
    alarm_off_t
    }alarm_actuator_id;

void alarm_actuator_init();
state_define(alarm_waiting);
state_define(alarm_off);
state_define(alarm_on);


#endif /* ALARM_ACTUATOR_H_ */

```

.c file

untitled

```
#include "alarm_actuator.h"
#define alarm_timer 32000000    //by some calculations this number is equivalent to 60sec. delay

void (*actuator_state_ptr)();
void alarm_actuator_init()
{
state_define(alarm_off)
    alarm_actuator_id = alarm_waiting_t;
    actuator_state_ptr = state(alarm_off);
}
state_define(alarm_waiting)
{
    alarm_actuator_id = alarm_waiting_t;

{
    alarm_actuator_id = alarm_off_t;
    Set_Alarm_actuator(1);                          //state action
    actuator_state_ptr = state(alarm_waiting);
}
state_define(alarm_on)
{
    alarm_actuator_id = alarm_on_t;
    Set_Alarm_actuator(0);                           //state action
    Delay(alarm_timer);
    Set_Alarm_actuator(1);                           //state action
    actuator_state_ptr = state(alarm_waiting);
}
void set_alarm_on()
{
    actuator_state_ptr = state(alarm_on);

}
void set_alarm_off()
{
    actuator_state_ptr = state(alarm_off);

}

```

- States.h

```c
/*
 * Sates.h
 *
 *  Created on: Oct 15, 2023
 *      Author: ahmed
 */

#ifndef STATES_H_
#define STATES_H_

#include <stdio.h>

// functions definitions
#define state_define(_func_) void _func_()
#define state(_func_) _func_


// interfaces functions between modules
void send_pressure_value(int value);
void high_pressure_detected();
void set_alarm_on();
void set_alarm_off();

#endif /* STATES_H_ */

```

- App.c

```c
/*
 ============================================================================
 Name        : Pressure_Detection.c
 Author      : Eng.Ahmed Nabil Mahmoud
 Version     :
 Copyright   :
 Description : high pressure detection
 ============================================================================
 */

#include "pressure_sensor.h"
#include "main_algorithm.h"
#include "alarm_control.h"
#include "alarm_actuator.h"

extern void (*sensor_state_ptr)();
extern void (* main_state_ptr)();
extern void (*alarm_control_ptr)();
extern void (*actuator_state_ptr)();


void setup()
{
      GPIO_INITIALIZATION ();
    sensor_init();
    alarm_actuator_init();
    }
int main()
{
    setup();
    while(1)
    {
        sensor_state_ptr();
        main_state_ptr();
        alarm_control_ptr();
        actuator_state_ptr();
    }
}

```

8. Building_Files

- startup.c

```c
/*
 * startup.c
 *
 *  Created on: Oct 15, 2023
 *      Author: ahmed
 */
extern int main();
extern unsigned int E_text;
extern unsigned int S_data;                 //symbols defined in liker_Script
extern unsigned int E_data;
extern unsigned int S_bss;
extern unsigned int E_bss;
extern unsigned int Stack_top;


void reset_handler();
void default_handler();
void NMI_handler()__attribute__((weak,alias("default_handler")));;
void H_FAULT_handler()__attribute__((weak,alias("default_handler")));;        //interrupt vector table
void MM_handler()__attribute__((weak,alias("default_handler")));;
void BUS_handler()__attribute__((weak,alias("default_handler")));;
void USAGE_FAULT_handler()__attribute__((weak,alias("default_handler")));;



    void(*vector_table[])()__attribute__((section(".vector"))) =
    {
            (void(*)())&Stack_top,
            reset_handler,
            NMI_handler,
            H_FAULT_handler,                                    //array of pointer to function
            MM_handler,
            BUS_handler,
            USAGE_FAULT_handler
    };

    void reset_handler()
    {
        unsigned int i;
        unsigned int data_size = (unsigned char*)&E_data - (unsigned char*)&S_data ;
        unsigned char* target = (unsigned char*)&S_data;
        unsigned char* base = (unsigned char*)&E_text;
        for(i=0; i<data_size; i++)
        {
            *target++ = *base++;                            //copying .data sec from flash to ram
        }
        unsigned int bss_size = (unsigned char*)&E_bss - (unsigned char*)&S_bss ;
        unsigned char* target_b = (unsigned char*)&S_bss;
        for(i=0; i<bss_size; i++)
        {
            *target_b++ = 0;                                //reserve .bss section by 0
        }
        main();
    }
    void default_handler()
    {
        reset_handler();
    }

```

- linker_script

```
/* linker_scirpt -> ARM cortex-m3
Eng. : Ahmed Nabil
*/

MEMORY
{
  FLASH (rx) : ORIGIN = 0X08000000 , LENGTH = 64K
  SRAM  (rwx) : ORIGIN = 0X20000000 , LENGTH = 20K
}

SECTIONS
{

  .text :
{

   *(.vector)
   *(.text)
    . = ALIGN(4) ;
     *(.rodata)
    . = ALIGN(4) ;
    E_text = . ;

}> FLASH


.data :
{

    S_data = . ;
    *(.data)
    . = ALIGN(4) ;
    E_data = . ;

}> SRAM AT> FLASH

.bss :
  {
     S_bss = . ;
     *(.bss)
     . = ALIGN(4) ;
     *(.common)
     . = ALIGN(4) ;
     E_bss = . ;

  }
  . = . + 0x2000 ;
  Stack_top = . ;
}
```

- makefile

```
CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -mthumb -gdwarf-2
INCS=-I .
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
project_name=Pressure_Detection

all: $(project_name).bin
    @echo --build is done--

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

$(project_name).elf: $(OBJ)
    $(CC)ld.exe -T linker_script.ld $(OBJ) -o $@ -Map=Map_file.map

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



clean:
    rm *.elf *.bin
clean_all:
    rm *.o *.elf *.bin *.hex *.asm
```

9. Software_analysis

- .map file

```
Allocating common symbols
Common symbol       size              file

main_state_id       0x1               app.o
alarm_actuator_id   0x1               app.o
sensor_state_id     0x1               pressure_sensor.o
actuator_state_ptr  0x4               alarm_actuator.o
alarm_control_id    0x1               app.o
Pressure_value      0x4               pressure_sensor.o
sensor_state_ptr    0x4               pressure_sensor.o
pressure_value_m    0x4               main_algorithm.o

Memory Configuration

Name             Origin             Length             Attributes
FLASH            0x08000000         0x00010000         xr
SRAM             0x20000000         0x00005000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map


.text           0x08000000      0x478
 *(.vector)
 .vector        0x08000000       0x1c startup.o
                0x08000000                vector_table
 *(.text)
 .text          0x0800001c       0xbc startup.o
                0x0800001c                reset_handler
                0x080000cc                BUS_handler
                0x080000cc                USAGE_FAULT_handler
                0x080000cc                H_FAULT_handler
                0x080000cc                default_handler
                0x080000cc                MM_handler
                0x080000cc                NMI_handler
 .text          0x080000d8       0xac pressure_sensor.o
                0x080000d8                sensor_init
                0x08000110                reading
                0x08000148                waiting
 .text          0x08000184       0x50 app.o
                0x08000184                setup
                0x08000198                main
 .text          0x080001d4       0x48 main_algorithm.o
```

- Symbols table

```
$ arm-none-eabi-nm.exe Pressure_Detection.elf
08000340 T acting
20000020 B actuator_state_ptr
20000019 B alarm_actuator_id
08000388 T alarm_actuator_init
2000001a B alarm_control_id
20000008 D alarm_control_ptr
080003cc T alarm_off
080003fc T alarm_on
080003b4 T alarm_waiting
080000cc W BUS_handler
080001d4 T checking
080000cc T default_handler
0800021c T Delay
2000000c B E_bss
2000000c D E_data
08000478 T E_text
08000240 T getPressureVal
080002a8 T GPIO_INITIALIZATION
080000cc W H_FAULT_handler
0800036c T high_pressure_detected
08000328 T idle
08000198 T main
20000018 B main_state_id
20000004 D main_state_ptr
080000cc W MM_handler
080000cc W NMI_handler
20000010 B Pressure_value
2000001c B pressure_value_m
08000110 T reading
0800001c T reset_handler
2000000c B S_bss
20000000 D S_data
080001fc T send_pressure_value
080000d8 T sensor_init
20000000 D sensor_pull_time
2000000c B sensor_state_id
20000014 B sensor_state_ptr
08000258 T Set_Alarm_actuator
0800045c T set_alarm_off
08000440 T set_alarm_on
08000184 T setup
20002024 B Stack_top
080000cc W USAGE_FAULT_handler
08000000 T vector_table
08000148 T waiting
```

- Sections table

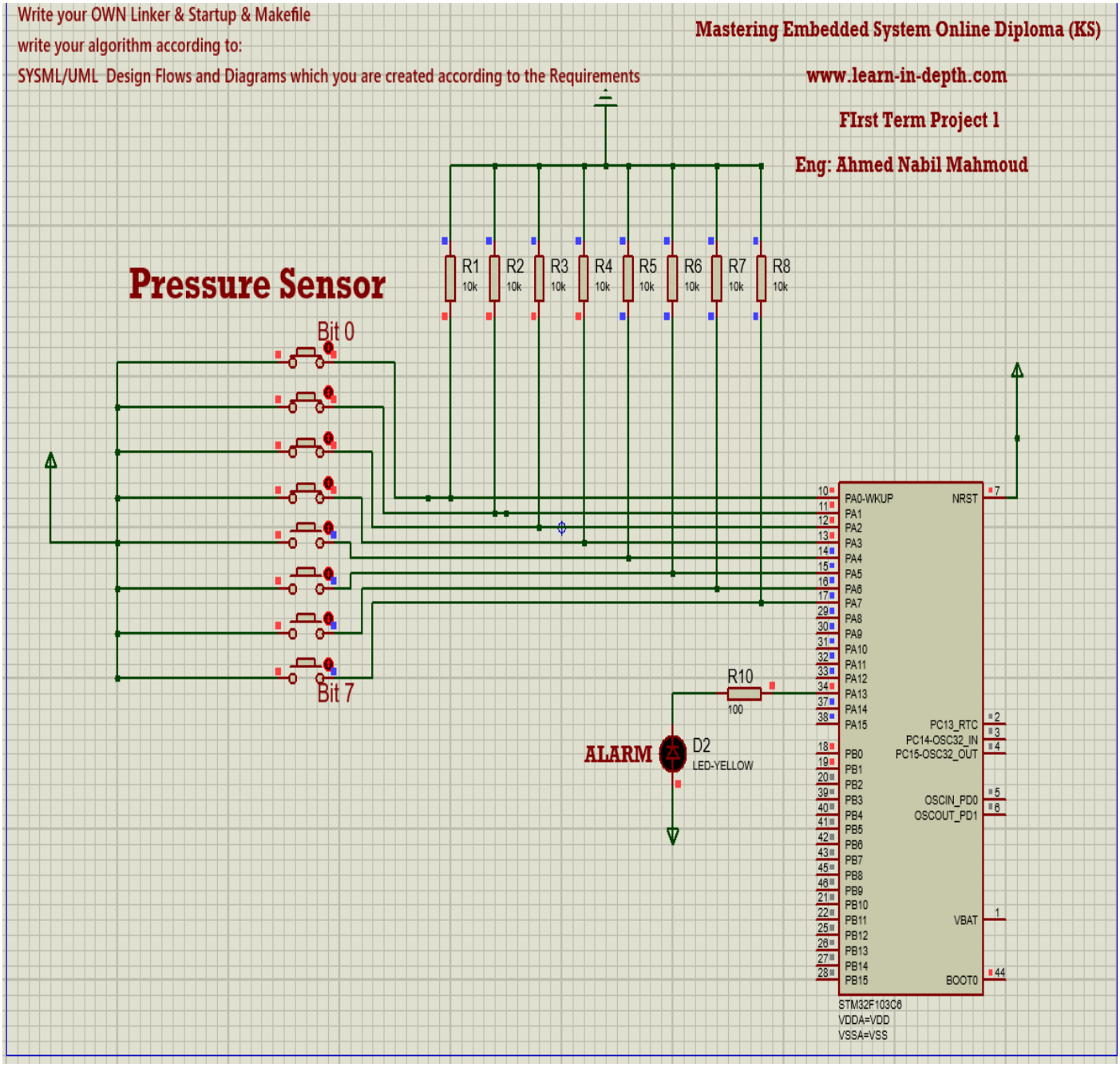
```
$ arm-none-eabi-objdump.exe -h Pressure_Detection.elf

Pressure_Detection.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000478  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         0000000c  20000000  08000478  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000018  2000000c  2000000c  0001000c  2**2
                  ALLOC
  3 .debug_info   00000837  00000000  00000000  0001000c  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 00000451  00000000  00000000  00010843  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000003f8  00000000  00000000  00010c94  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000e0  00000000  00000000  0001108c  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   0000032d  00000000  00000000  0001116c  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    00000355  00000000  00000000  00011499  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  000117ee  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000117ff  2**0
                  CONTENTS, READONLY
 11 .debug_frame  000002bc  00000000  00000000  00011834  2**2
                  CONTENTS, READONLY, DEBUGGING
```

10. Simulation

- Pressure less than threshold



- Pressure more than threshold