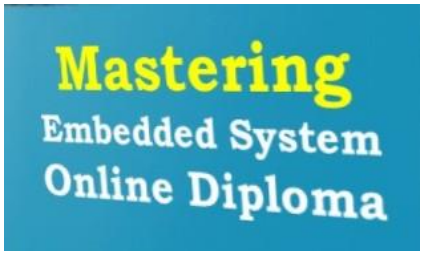


Pressure Detection Project

Mastering Embedded Systems Diploma

www.learn-in-depth.com

First Term (Final Project 1)

Eng. Amr Esaam Mohammed Zidan

My Profile: https://www.learn-in-depth.com/online-diploma/amresaam1342000%40gmail.com

Contents

1. Case Study

• Requirements

The client requires a system with the following specifications:

1. Pressure controller informing the cabin crew when pressure exceeds 20 bars.
2. The informing method is an alarm operating for 60 seconds.
3. Keep track of the measured values (optional).

• Assumptions

The following assumptions are made:

1. No setup or shut down for the microcontroller.
2. No maintenance for the microcontroller.
3. Neither the pressure sensor nor the alarm fails
4. No power cuts for the microcontroller

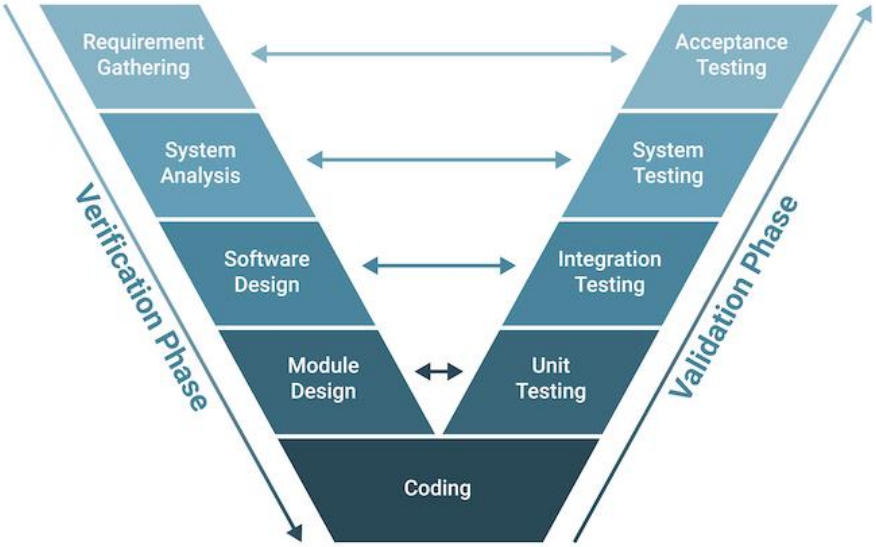
• Versioning

The possibility of storing pressure sensor readings might be included in a future version.

2. Method

Software Development Lifecycle and Software Testing Lifecycle

The (SDLC) and (STLC) will be approached based on the V-Model.



Requirements Gathering and Analysis: The first phase of the V-Model is the requirements gathering and analysis phase, where the customer's requirements for the software are gathered and analyzed to determine the scope of the project.

Design: In the design phase, the software architecture and design are developed, including the high-level design and detailed design.

Implementation: In the implementation phase, the software is actually built based on the design.

Testing: In the testing phase, the software is tested to ensure that it meets the customer's requirements and is of high quality.

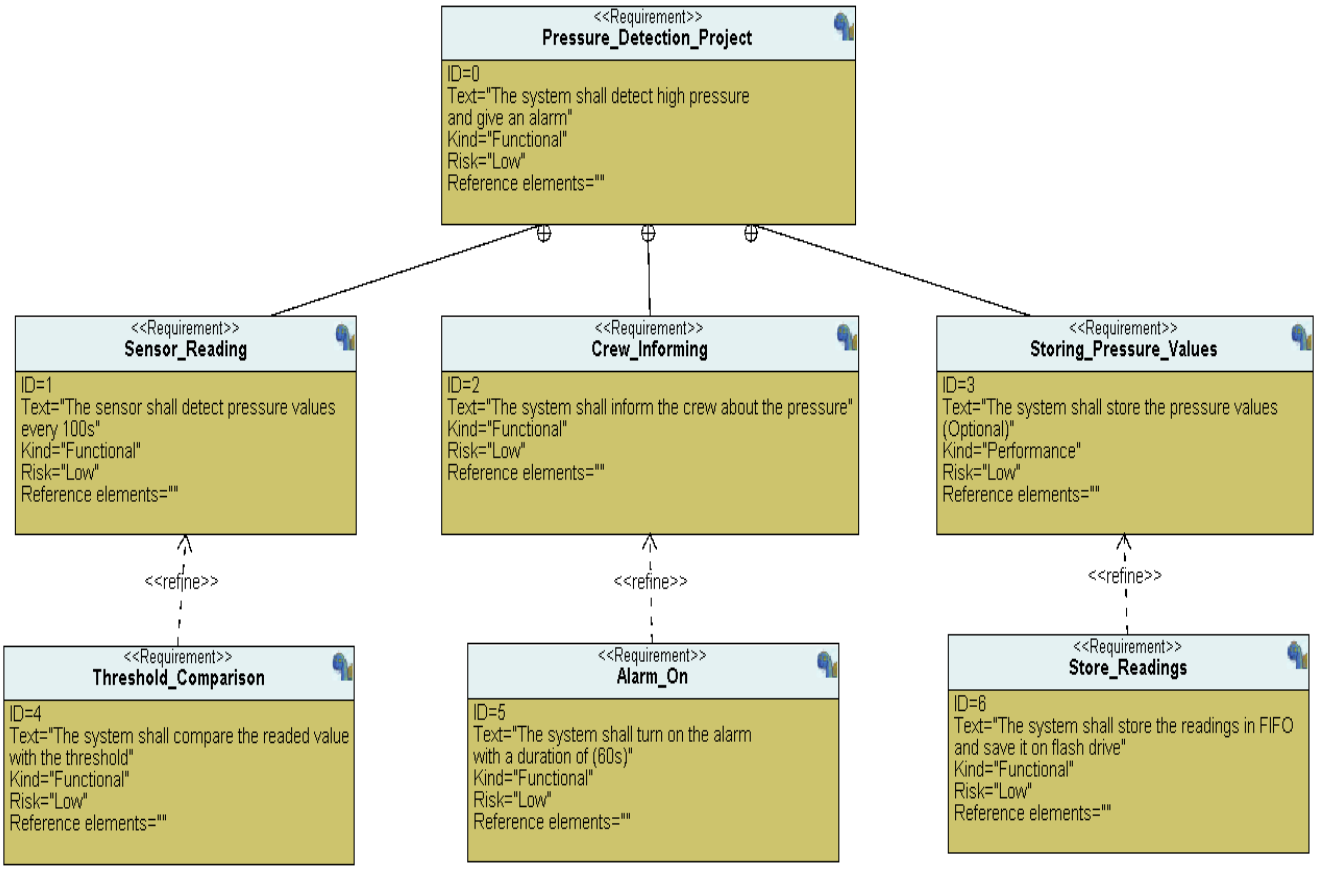
Deployment: In the deployment phase, the software is deployed and put into use.

Maintenance: In the maintenance phase, the software is maintained to ensure that it continues to meet the customer's needs and expectations.

The V-Model is often used in safety-critical systems, such as aerospace and defense systems, because of its emphasis on thorough testing and its ability to clearly define the steps involved in the software development process.

3. System Requirements

Requirement model



4. Design Space Exploration

Microcontroller: one stm32f103c8t6 SoC will be used as it meets all the needed technical requirements such as: suitable processor, acceptable flash memory size and small size as well as being cost efficient.

Overview: The STM32F103C8T6 is a medium density performance line, ARM Cortex-M3 32bit microcontroller in 48 pin LQFP package. It incorporates high performance RISC core with 72MHz operating frequency, high speed embedded memories, extensive range of enhanced I/Os and peripherals connected to two APB buses. The STM32F103C8T6 features 12bit ADC, timers, PWM timer, standard and advanced communication interfaces. A comprehensive set of power saving mode allows the design of low power applications.

Features:

Operating voltage range from 2V to 3.6V

64Kbytes of flash memory

20Kbytes of SRAM

CRC calculation unit, 96bit unique ID

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

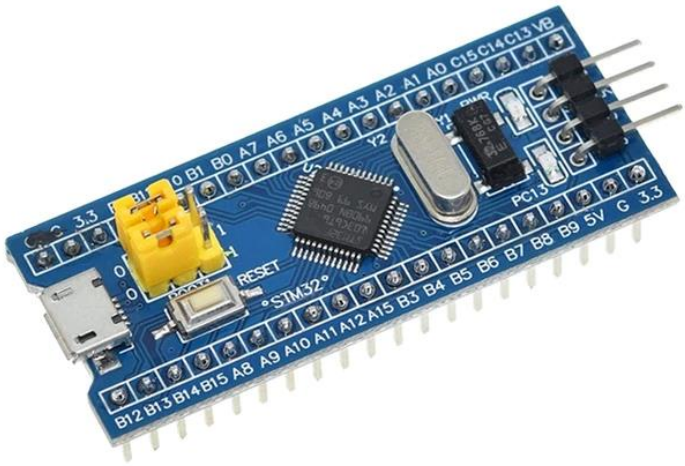
7 channel DMA controller, 3 general purpose timer and 1 advanced control timer

37 fast I/O ports

Serial wire debug (SWD) and JTAG interfaces

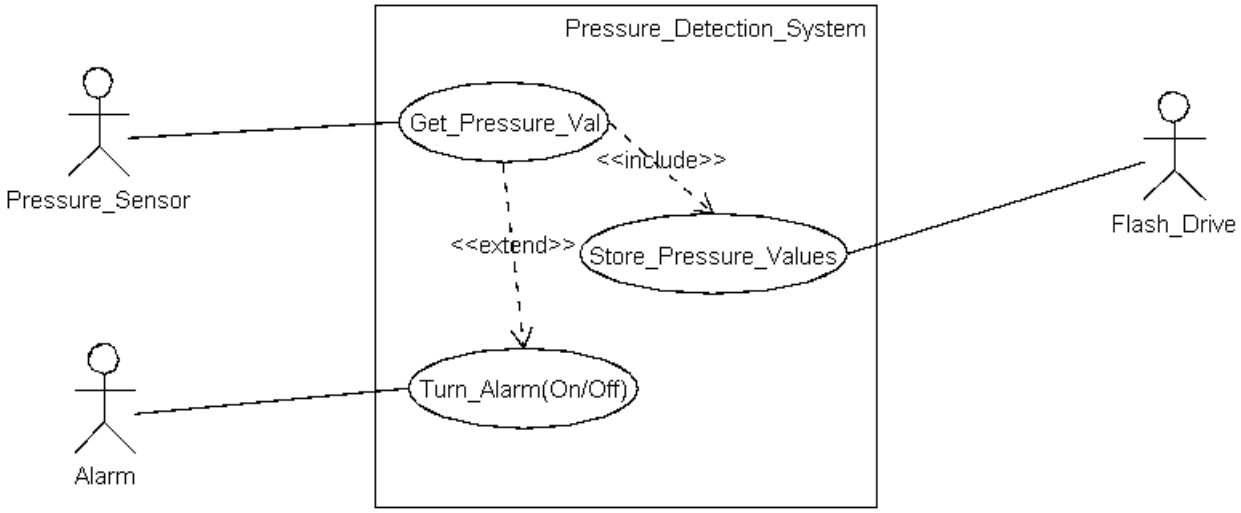
Two SPI, two I2C, three USART, one USB and one CAN interfaces

Ambient operating temperature range from -40°C to 85°C

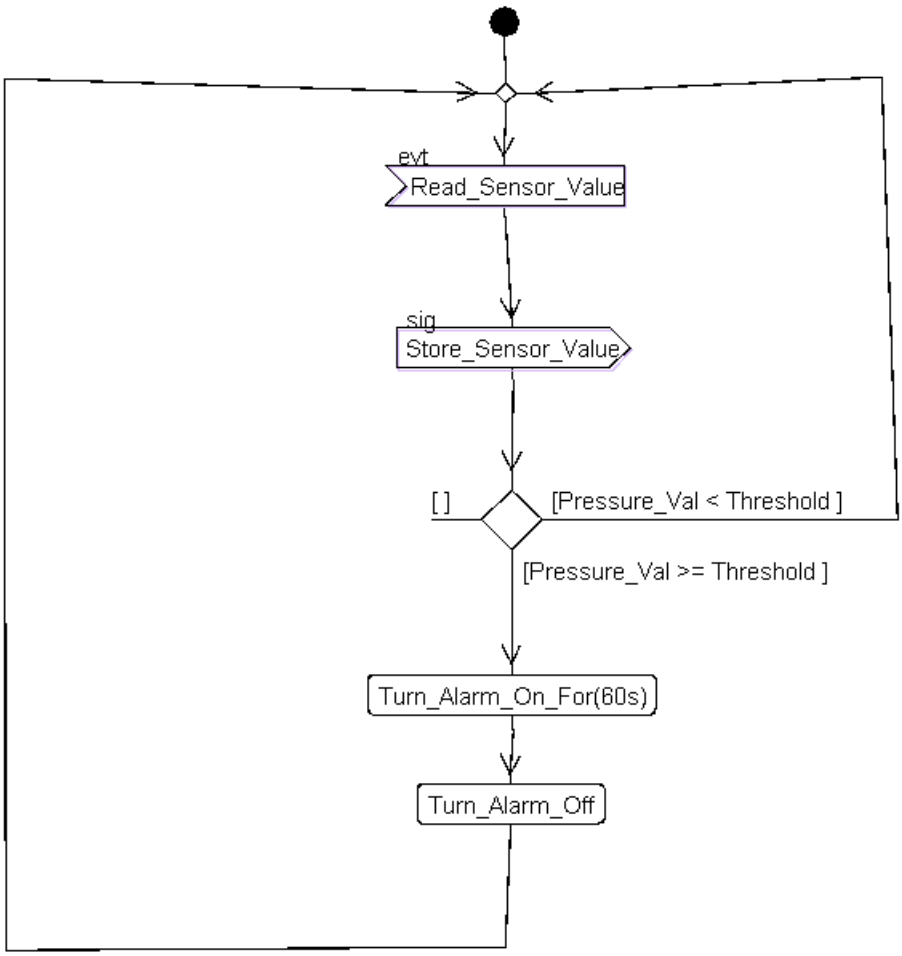


5. System Analysis

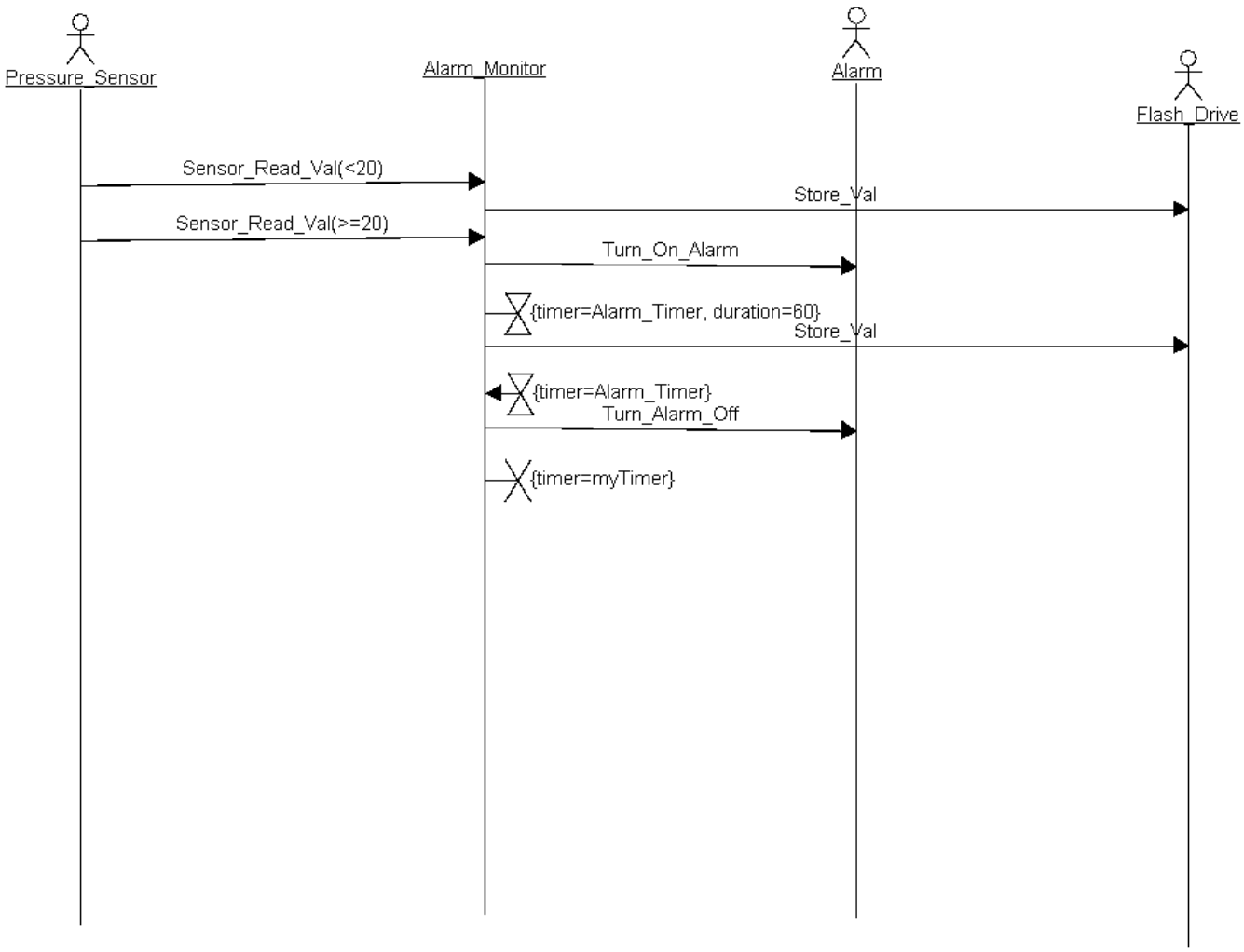
- Use Case Diagram



- Activity Diagram

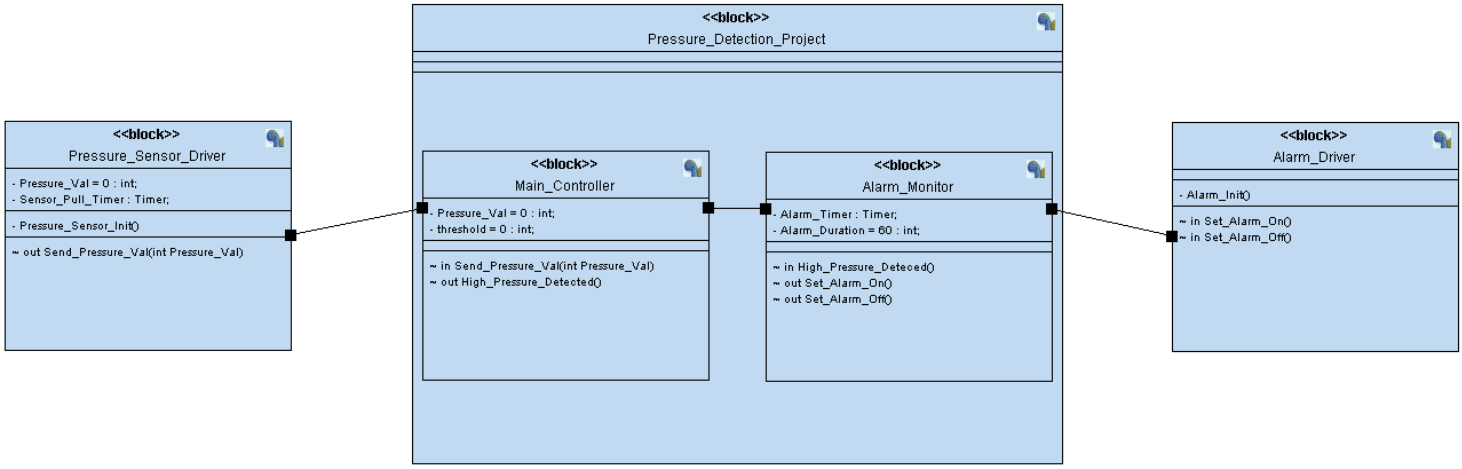


- Sequence Diagram



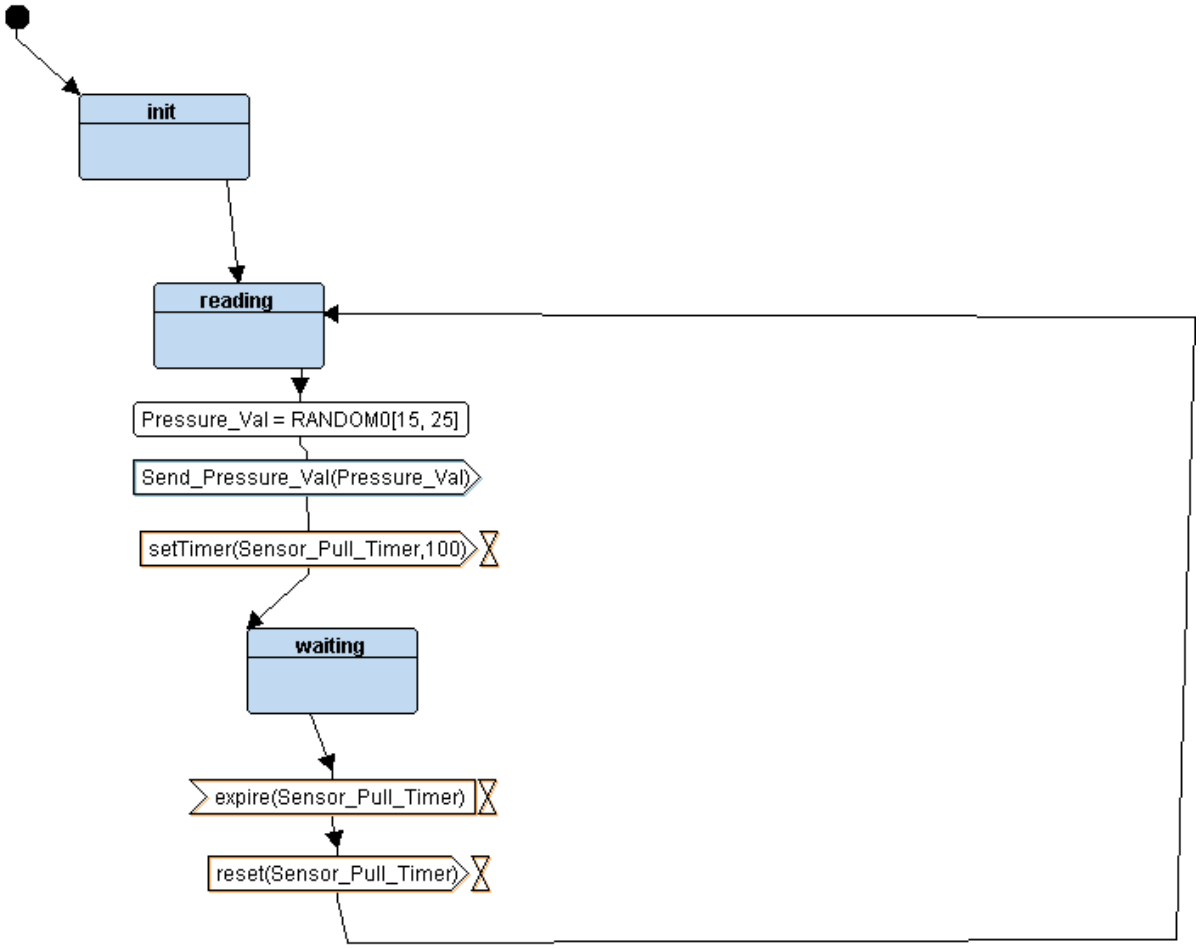
6. System Design

- Block Diagrams

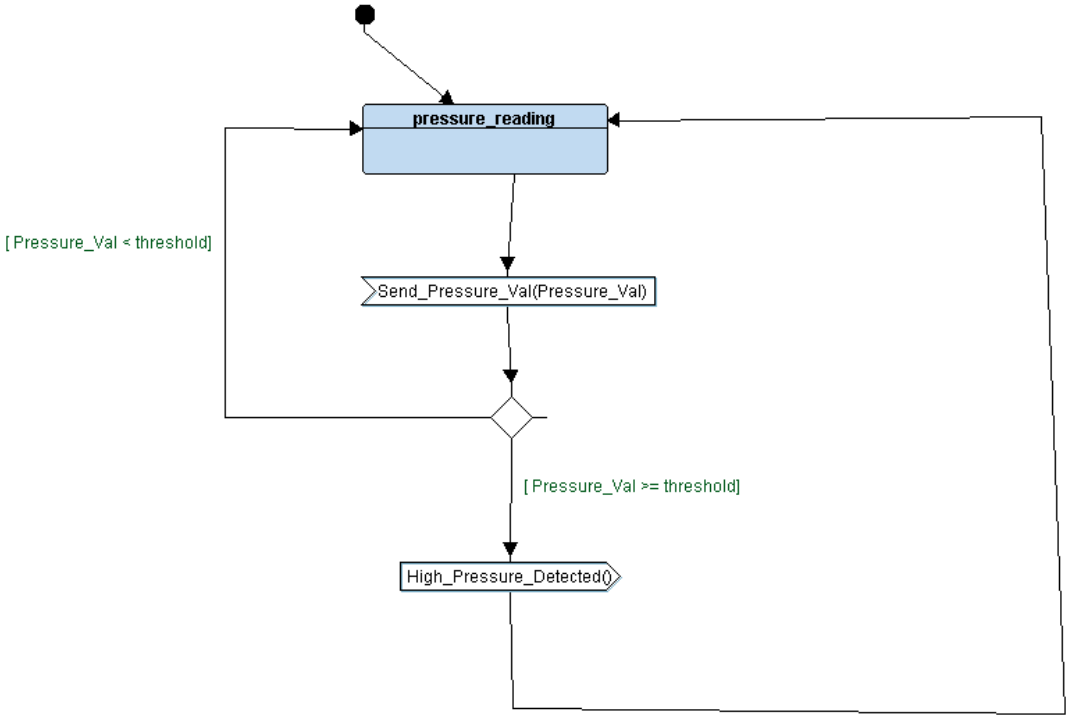


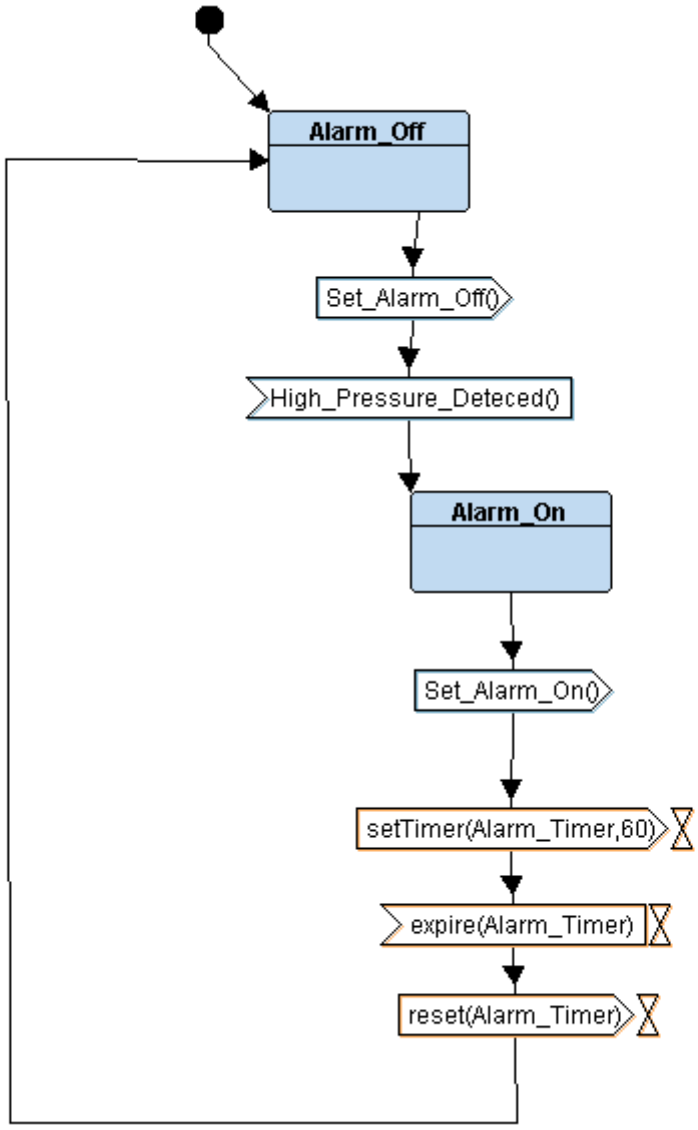
- State Machine Diagrams

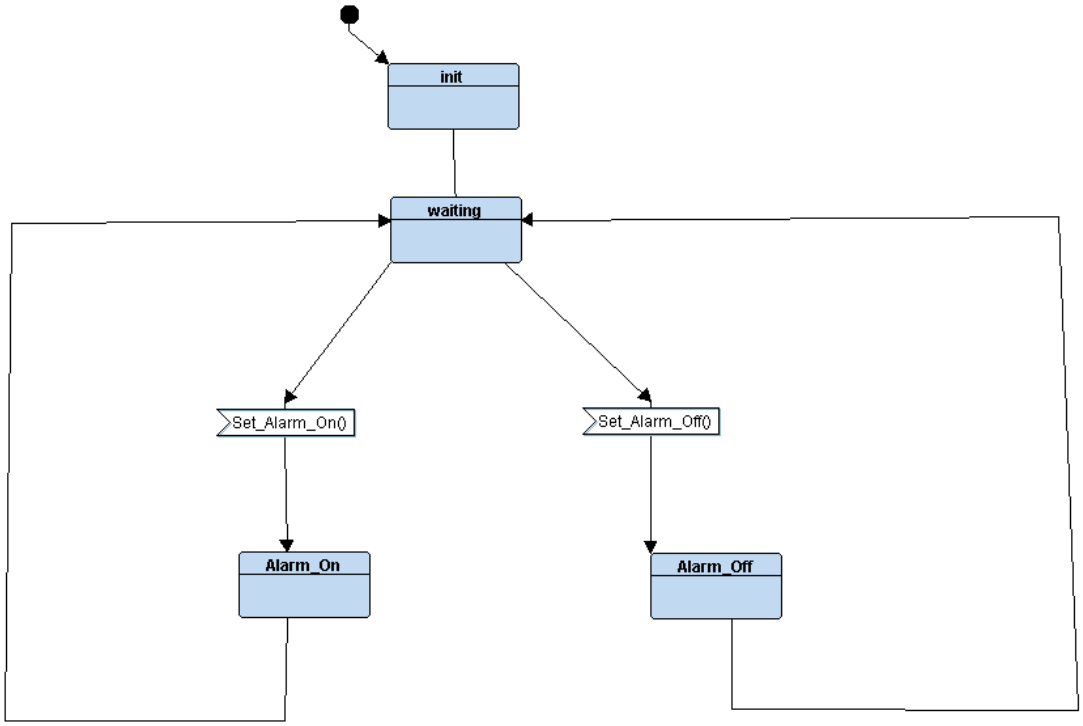
Pressure Sensor



Main Controller




Alarm_Off
Set_Alarm_Off()
High_Pressure_Deteced()
Alarm_On
Set_Alarm_On()
setTimer(Alarm_Timer,60)
expire(Alarm_Timer)
reset(Alarm_Timer)


init
waiting
Set_Alarm_On()
Set_Alarm_Off()
Alarm_On
Alarm_Off

- C Codes

Pressure Sensor

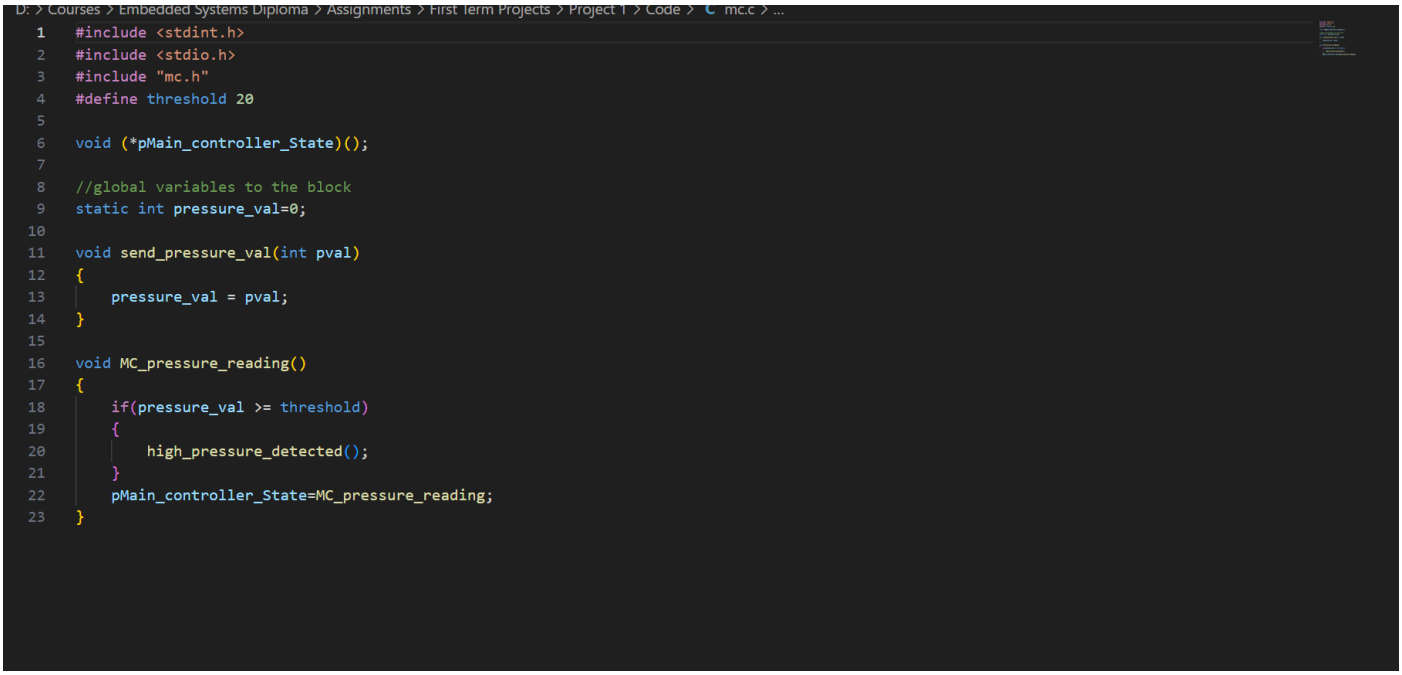
.c file

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



//global variables to the block
static int pressure_val=0;

//state pointer (ptr to function)
void (*pPressure_sensor_State)();



//init pressure_sensor driver
void pressure_sensor_init()
{
    //init
}


//reading state
STATE_DEFINE(pressure_sensor_reading)
{
    //state id
    pressure_sensor_state_id=pressure_sensor_reading_state;

     //state action
    pressure_val=getPressureVal();
    send_pressure_val(pressure_val);

```

.h file

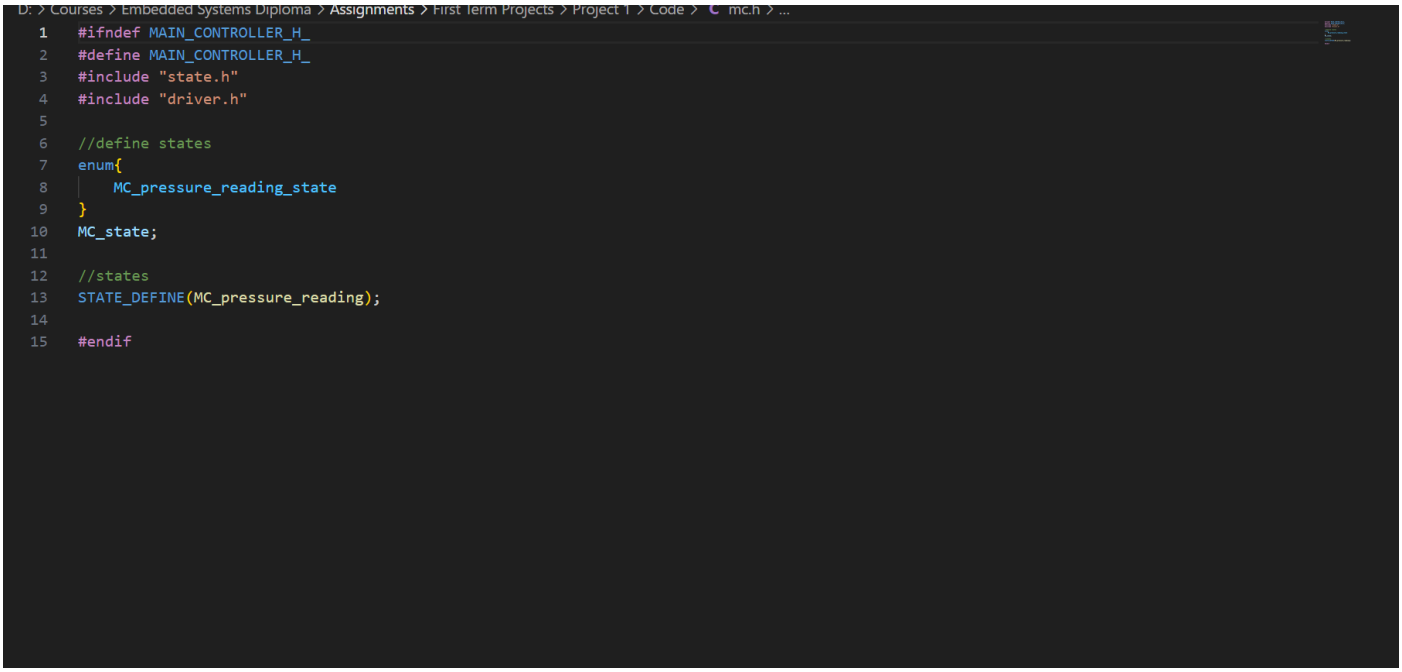
```c
#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_

#include "state.h"
#define sensor_pull_time 100

//define states
enum {
    pressure_sensor_reading_state,
    pressure_sensor_waiting_state
}pressure_sensor_state_id;

//states
STATE_DEFINE(pressure_sensor_reading);
STATE_DEFINE(pressure_sensor_waiting);

void pressure_sensor_init();

#endif
```

.c file



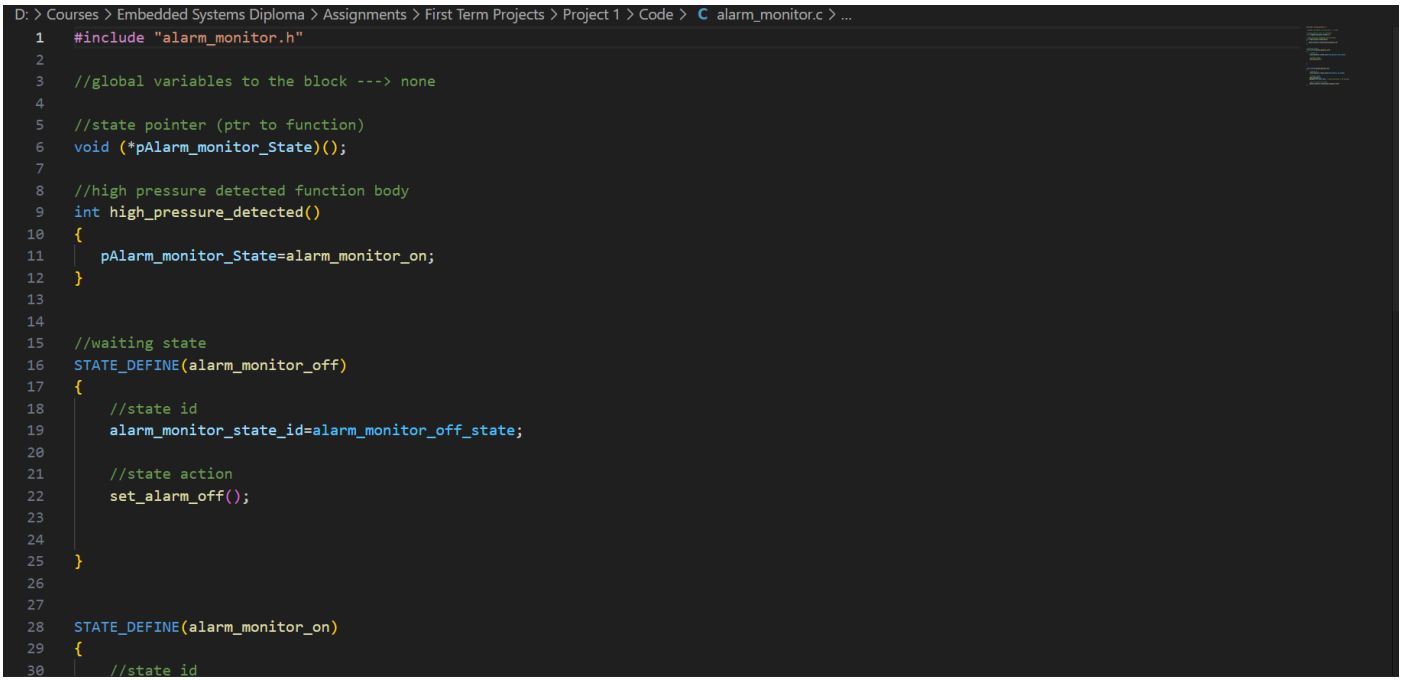
```c
#include <stdint.h>
#include <stdio.h>
#include "mc.h"
#define threshold 20

void (*pMain_controller_State)();

//global variables to the block
static int pressure_val=0;

void send_pressure_val(int pval)
{
    pressure_val = pval;
}

void MC_pressure_reading()
{
    if(pressure_val >= threshold)
    {
        high_pressure_detected();
    }
    pMain_controller_State=MC_pressure_reading;
}
```

.h file

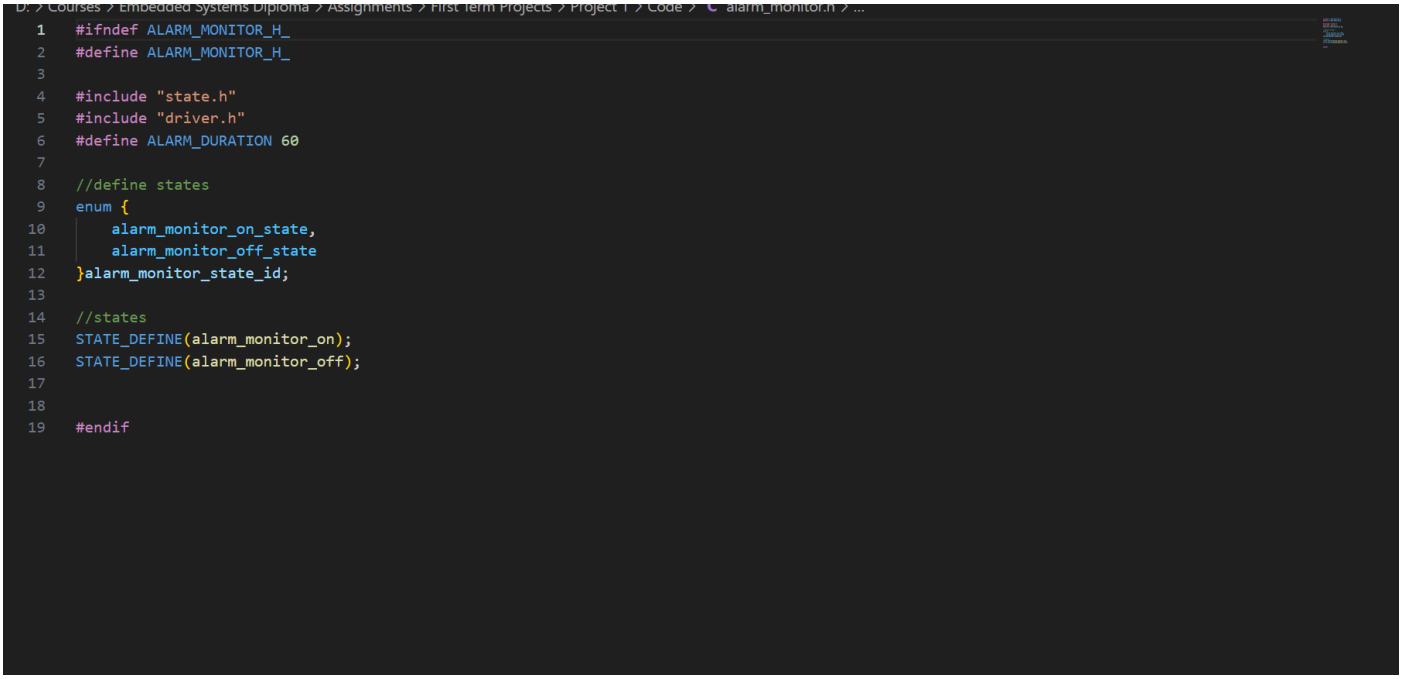


```c
#ifndef MAIN_CONTROLLER_H_
#define MAIN_CONTROLLER_H_
#include "state.h"
#include "driver.h"

//define states
enum{
    MC_pressure_reading_state
}
MC_state;

//states
STATE_DEFINE(MC_pressure_reading);

#endif
```

.c file


D: > Courses > Embedded Systems Diploma > Assignments > First Term Projects > Project 1 > Code > C alarm_monitor.c > ...
#include "alarm_monitor.h"
//global variables to the block ---> none
//state pointer (ptr to function)
void (*pAlarm_monitor_State)();
//high pressure detected function body
int high_pressure_detected()
{
pAlarm_monitor_State=alarm_monitor_on;
}
//waiting state
STATE_DEFINE(alarm_monitor_off)
{
//state id
alarm_monitor_state_id=alarm_monitor_off_state;
//state action
set_alarm_off();
}
STATE_DEFINE(alarm_monitor_on)
{
//state id

.h file


#ifndef ALARM_MONITOR_H_
#define ALARM_MONITOR_H_
#include "state.h"
#include "driver.h"
#define ALARM_DURATION 60
//define states
enum {
alarm_monitor_on_state,
alarm_monitor_off_state
}alarm_monitor_state_id;
//states
STATE_DEFINE(alarm_monitor_on);
STATE_DEFINE(alarm_monitor_off);
#endif

.c file

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

//global variables to the block ---> none

//state pointer (ptr to function)
void (*pAlarm_State)();


//init alarm driver
void alarm_init()
{
    //init
}

//alarm on function body
void set_alarm_on()
{
    Set_Alarm_actuator(1);
}

//alarm off function body
void set_alarm_off()
{
    Set_Alarm_actuator(0);
}
```

.h file

```c
#ifndef ALARM_H_
#define ALARM_H_

#include "state.h"

//define states
enum {
    alarm_waiting_state,
    alarm_on_state,
    alarm_off_state
}alarm_state_id;

//states
STATE_DEFINE(alarm_waiting);
STATE_DEFINE(alarm_on);
STATE_DEFINE(alarm_off);

void alarm_init();

#endif
```

.c file

```c

void setup()
{
    //init all drivers
    //init IRQ
    //init US & DC_MOTOR
    //init block
    pressure_sensor_init();
    alarm_init();
    GPIO_INITIALIZATION();
    //Set state ptr for each block
    pPressure_sensor_State=STATE(pressure_sensor_reading);
    pMain_controller_State=STATE(MC_pressure_reading);
    pAlarm_monitor_State=STATE(alarm_monitor_off);
    pAlarm_State=STATE(alarm_waiting);
}


int main ()
{
    setup();
    while (1)
    {
        pPressure_sensor_State();
        pMain_controller_State();
        pAlarm_monitor_State();
        pAlarm_State();
    }
}
```

Code Building Tools

Startup code

D: > Courses > Embedded Systems Diploma > Assignments > First Term Projects > Project 1 > Code > startup.s

```
/* ARM Corex M3 Startup code
By: Eng. Amr Zidan */

/* SRAM from 0x20000000 */
.section .vectors
.word 0x20001000        /*stack top address*/
.word _reset            /*1 reset*/
.word _vector_handler   /*2 NMI*/
.word _vector_handler   /*3 Hard Fault*/
.word _vector_handler   /*4 MM Fault*/
.word _vector_handler   /*5 Bus Fault*/
.word _vector_handler   /*6 Usage Fault*/
.word _vector_handler   /*7 Reserved*/
.word _vector_handler   /*8 Reserved*/
.word _vector_handler   /*9 Reserved*/
.word _vector_handler   /*10 Reserved*/
.word _vector_handler   /*11 SV Call*/
.word _vector_handler   /*12 Debug Rserved*/
.word _vector_handler   /*13 Reserved*/
.word _vector_handler   /*14 Pend SV*/
.word _vector_handler   /*15 SysTick*/
.word _vector_handler   /*16 TRQ0*/
.word _vector_handler   /*17 TRQ1*/
.word _vector_handler   /*18 TRQ2*/
.word _vector_handler   /*19 ...*/

.section .text
_reset:
    bl main
    b .

.thumb_func

_vector_handler:
    b _reset
```

Linker Script

D: > Courses > Embedded Systems Diploma > Assignments > First Term Projects > Project 1 > Code > linker_script.ld

```
/* ARM Cortex M3 Linker Script
By: Eng. Amr Zidan */

MEMORY
{
    flash(RX) : ORIGIN = 0x08000000 , LENGTH = 128K
    sram(RWX) : ORIGIN = 0x20000000 , LENGTH = 20K
}

SECTIONS
{
    .text :
    {
        *(.vectors*)
        *(.text*)
        *(.rodata)
    }> flash

    .data :
    {
        *(.data*)
    }> flash

    .bss :
    {
        *(.bss*)
    }> sram

}
```

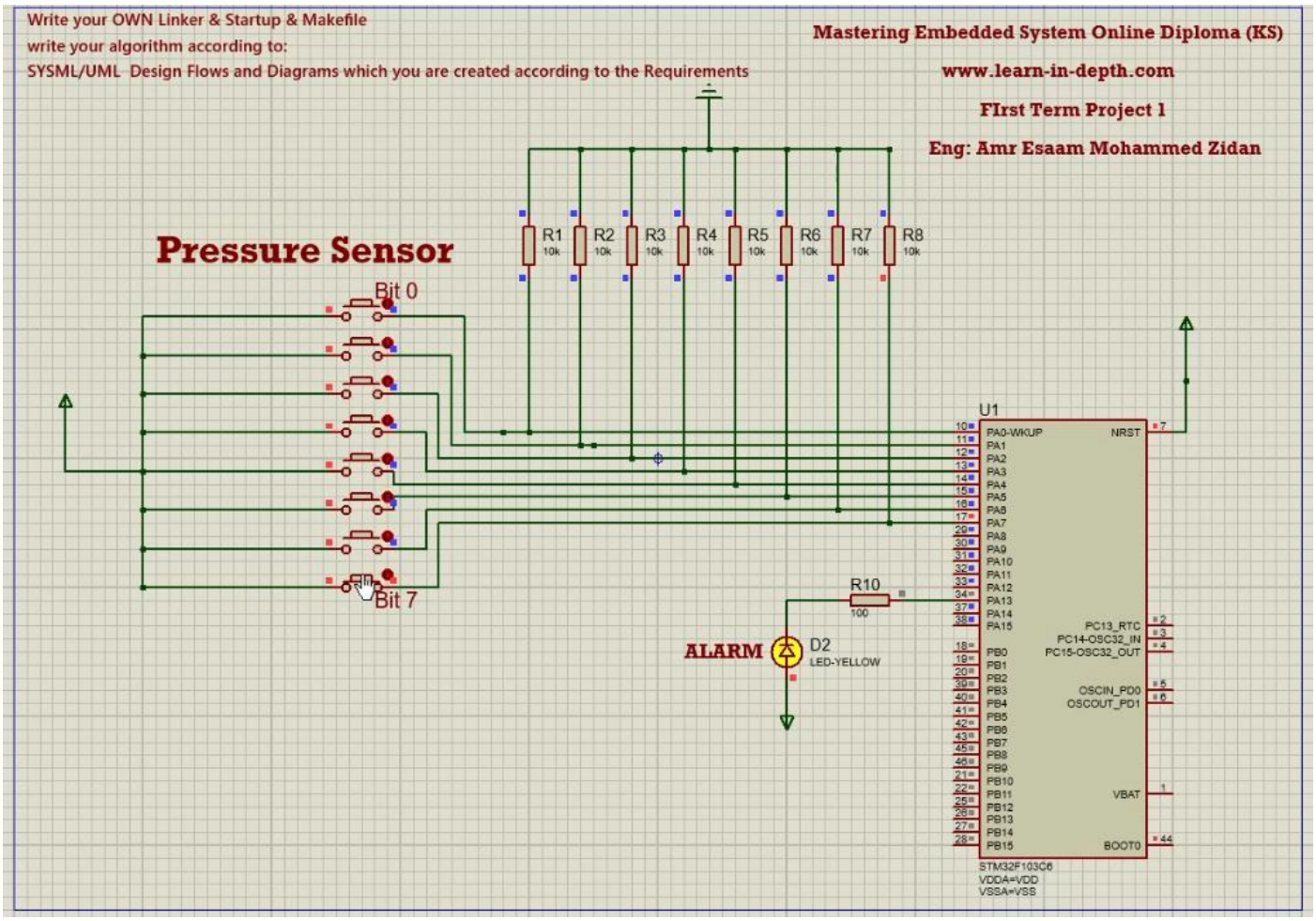
Makefile

D: > Courses > Embedded Systems Diploma > Assignments > First Term Projects > Project 1 > Code > makefile

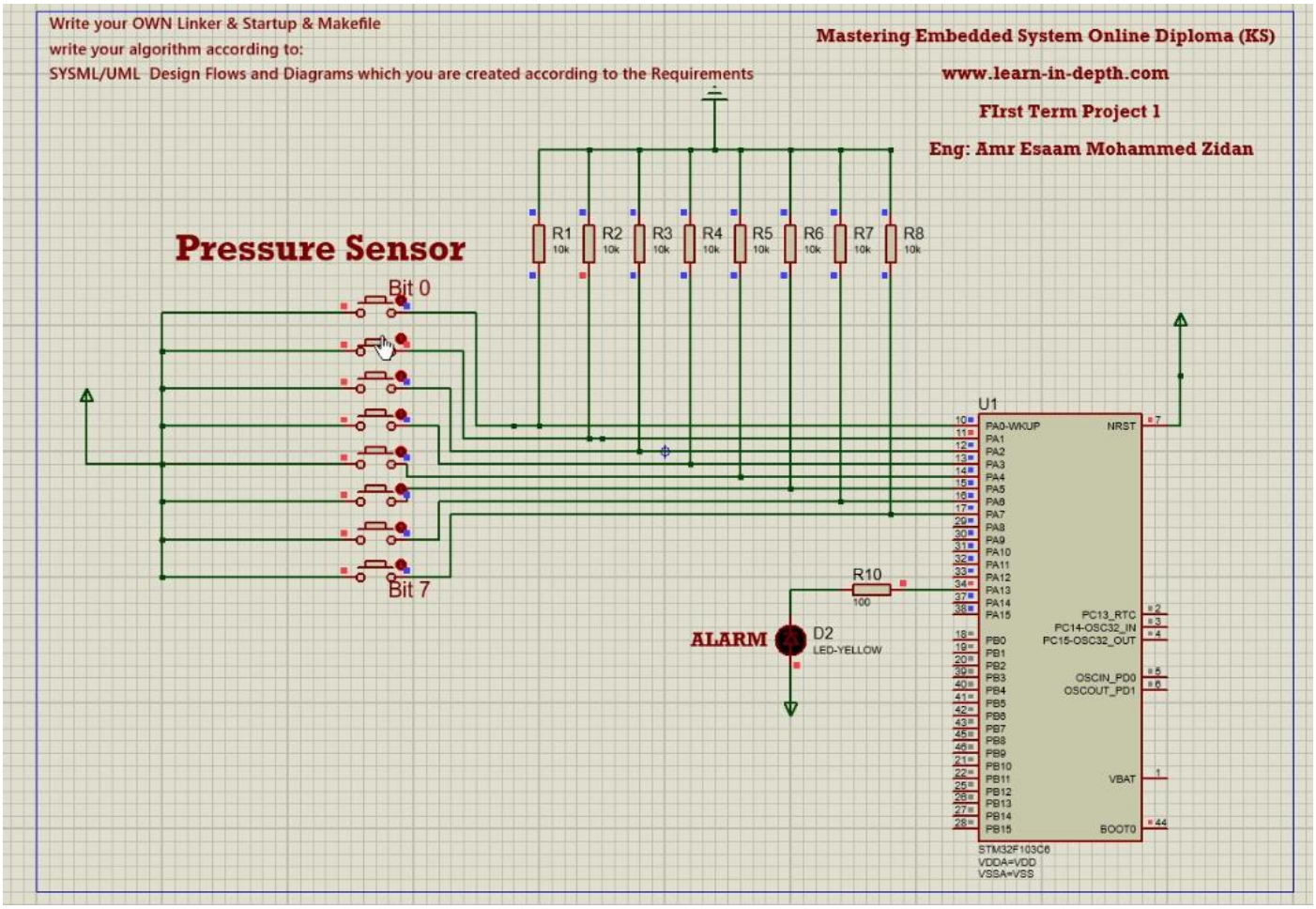
```
#copyrights: Amr Zidan

CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -mthumb -gdwarf-2
INCS= -I .
LIBS=
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
ASS=$(wildcard *.s)
ASSOBJ=$(ASS:.s=.o)
PROJECT_NAME=High_Pressure_Detection

all: $(PROJECT_NAME).bin
    @echo "-----Build is done-----"

startup.o: startup.s
    $(CC)as.exe $(CFLAGS) $< -o $@

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

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

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

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

clean:
    rm *.elf *.bin
```

- Simulation

Pressure equals or larger than threshold



Pressure less than threshold



.map file

```
Name             Origin             Length             Attributes
flash            0x08000000         0x00020000         xr
sram             0x20000000         0x00005000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map


.text           0x08000000      0x3d4
 *(.vectors*)
 .vectors       0x08000000       0x50 startup.o
 *(.text*)
 .text          0x08000050       0x9c alarm.o
                0x08000050                alarm_init
                0x0800005c                set_alarm_on
                0x0800006c                set_alarm_off
                0x0800007c                alarm_waiting
                0x08000094                alarm_on
                0x080000c0                alarm_off
 .text          0x080000ec       0x6c alarm_monitor.o
                0x080000ec                high_pressure_detected
                0x0800010c                alarm_monitor_off
                0x08000124                alarm_monitor_on
 .text          0x08000158      0x10c driver.o
                0x08000158                Delay
                0x0800017c                getPressureVal
                0x08000194                Set_Alarm_actuator
                0x080001e4                GPIO_INITIALIZATION
 .text          0x08000264       0x98 main.o
```

Symbol Table

```
$ arm-none-eabi-nm.exe High_Pressure_Detection.elf
080003cc t _reset
080003d2 t _vector_handler
08000050 T alarm_init
0800010c T alarm_monitor_off
08000124 T alarm_monitor_on
20000014 B alarm_monitor_state_id
080000c0 T alarm_off
08000094 T alarm_on
2000000c B alarm_state_id
0800007c T alarm_waiting
08000158 T Delay
0800017c T getPressureVal
080001e4 T GPIO_INITIALIZATION
080000ec T high_pressure_detected
080002c0 T main
0800031c T MC_pressure_reading
20000016 B MC_state
20000010 B pAlarm_monitor_State
20000008 B pAlarm_State
20000018 B pMain_controller_State
2000001c B pPressure_sensor_State
08000348 T pressure_sensor_init
08000354 T pressure_sensor_reading
20000015 B pressure_sensor_state_id
0800039c T pressure_sensor_waiting
20000004 b pressure_val
20000000 b pressure_val
080002fc T send_pressure_val
08000194 T Set_Alarm_actuator
0800006c T set_alarm_off
0800005c T set_alarm_on
08000264 T setup
```

Section Table

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000003d4  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .bss          00000020  20000000  20000000  00010000  2**2
                  ALLOC
  2 .debug_info   0000076d  00000000  00000000  000083d4  2**0
                  CONTENTS, READONLY, DEBUGGING
  3 .debug_abbrev 000003d1  00000000  00000000  00008b41  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_loc    00000394  00000000  00000000  00008f12  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_aranges 000000e0  00000000  00000000  000092a8  2**3
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_line   000002c2  00000000  00000000  00009388  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_str    0000037c  00000000  00000000  0000964a  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .comment      00000011  00000000  00000000  000099c6  2**0
                  CONTENTS, READONLY
  9 .ARM.attributes 00000031  00000000  00000000  000099d7  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000270  00000000  00000000  00009a08  2**2
                  CONTENTS, READONLY, DEBUGGING
```