Mastering Embedded system online diploma

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

Student(s)

Eng: BOULMANE Taha Omar

Profile learn-in-depth: https://www.learn-in-depth.com/online-diploma/boulmane818%40gmail.com

First term final project 1

Date of Submission

Saturday, August 7, 2021

Lecturer/Tutor

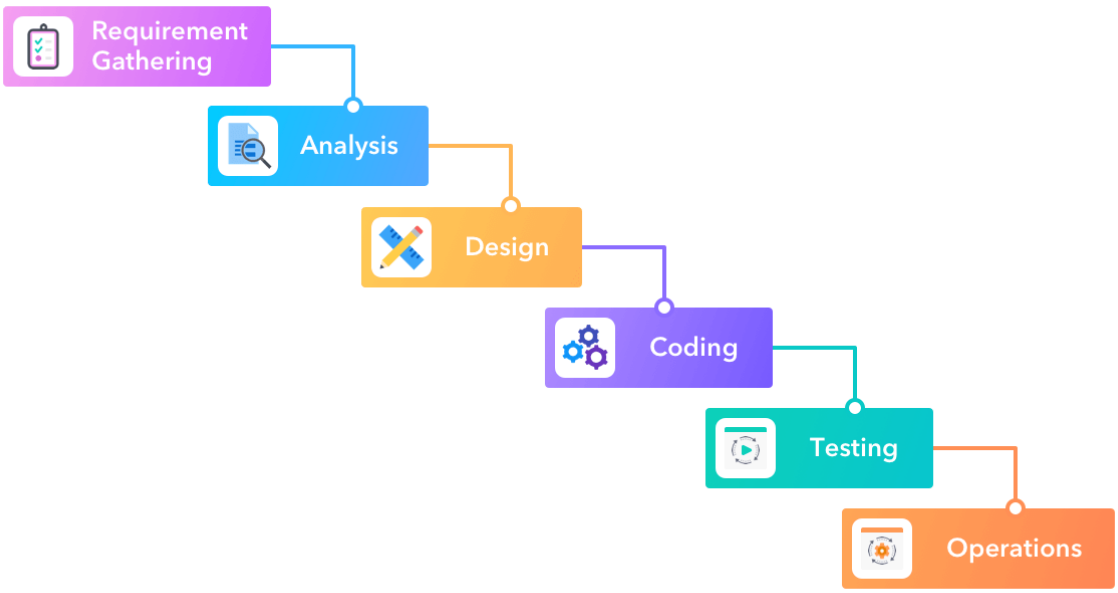
Keroles Shenoda

ABSTRACT

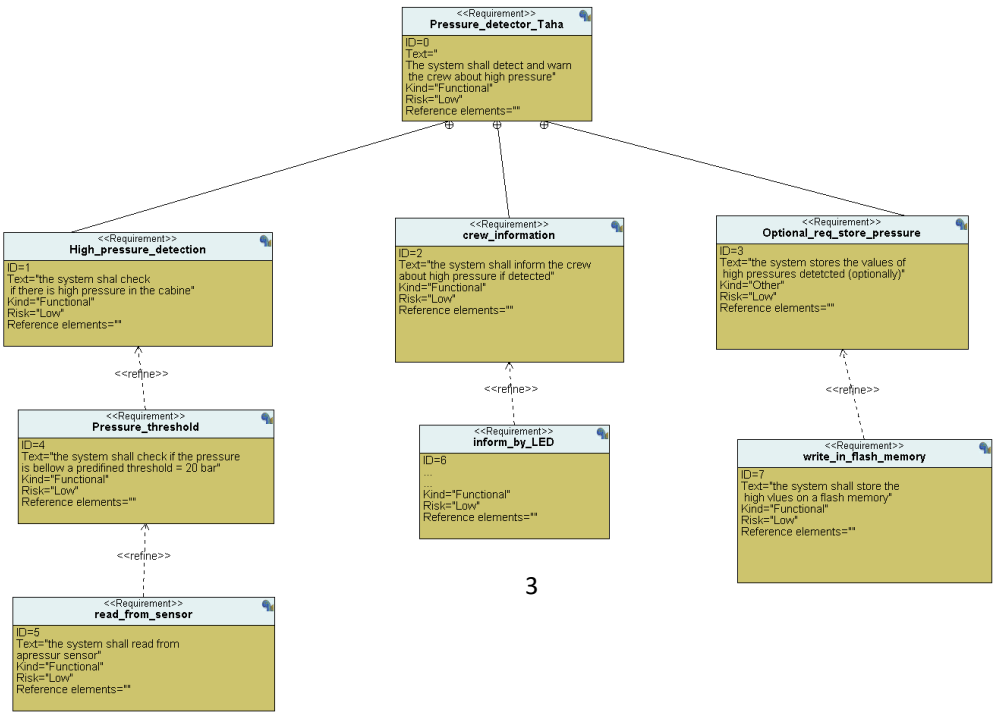
In this report we try to evaluate our competencies on the term of embedded C and system design using tools and knowledge we learned in depth in the first term of the learn-in-depth diploma. In this first project it's required that we design a system that inform pilots about high pressure if detected by lightning an LED for a delay.

INTRODUCTION

The first step in the system design is to choose in which way (methodology) you will proceed, in our case the system is simple enough that the waterfall method as explained in the figure below



In the phase of requirement gathering we made a meeting with the customer (Eng Keroles) and we get arranged about the following requirements

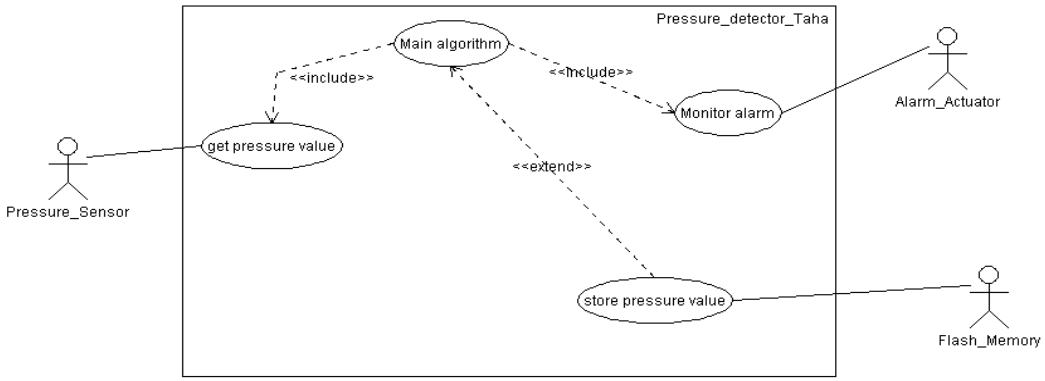


after that we analyzed the requirements the results of our analyzes lead to the diagrams described in the next section

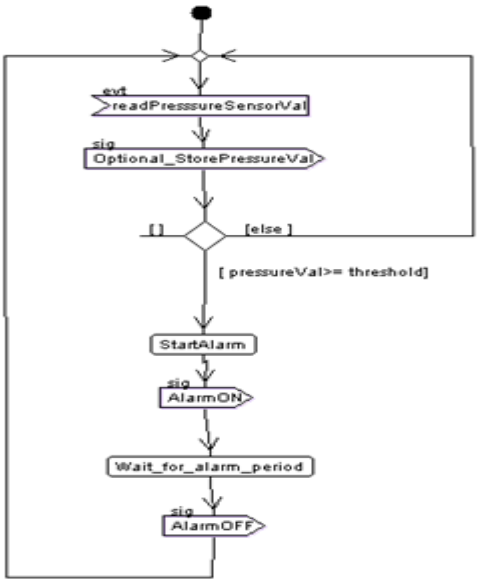
SYSTEM ANALYSIS

USE CASE DIAGRAM

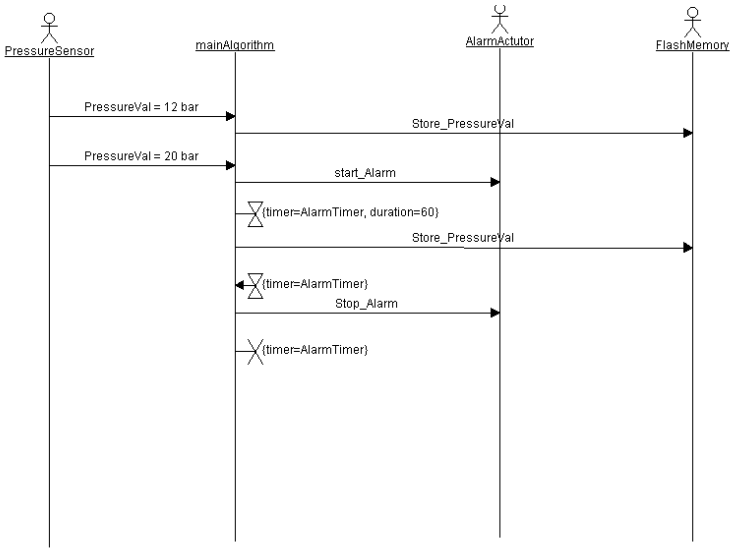
the figure below shows the use case diagram. The system communicate with three actors from the external environment (the third is optional)



.Activity diagram

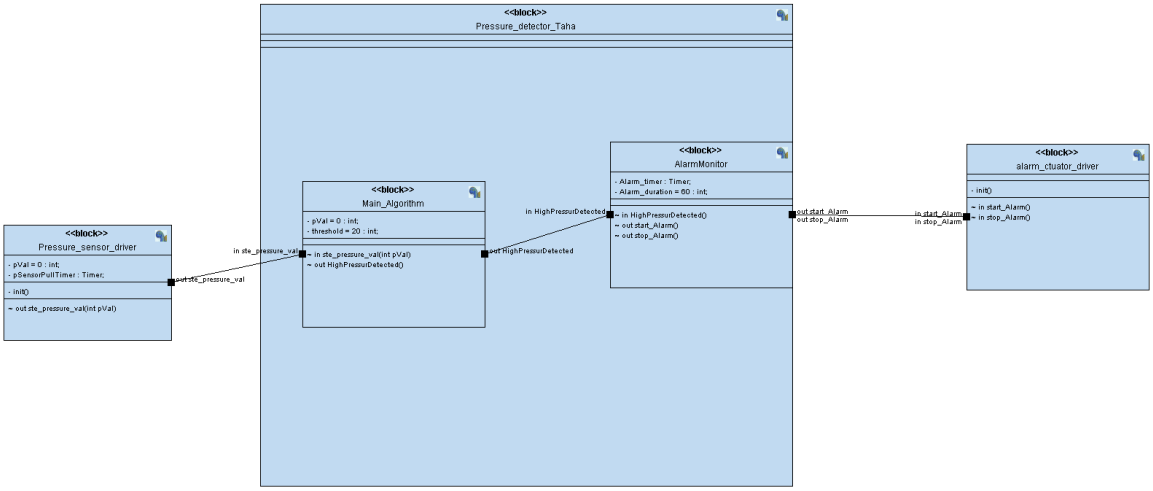


SEQUENCE DIAGRAM



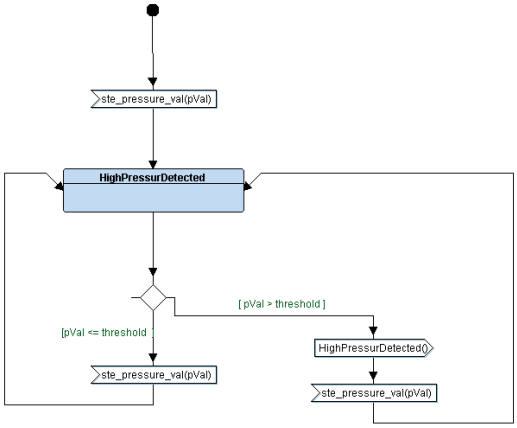
DESIGN :

our system will consist of three blocks as shown in the figure below

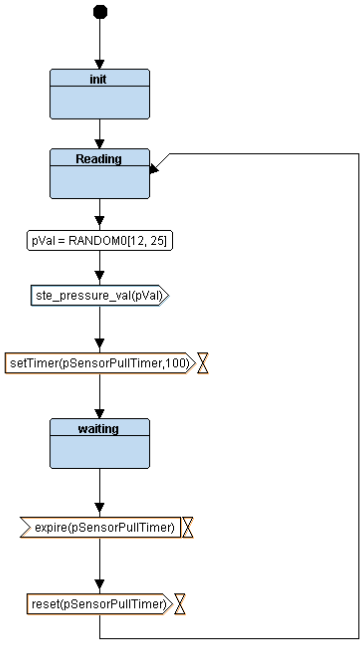


STATE MACHINES:

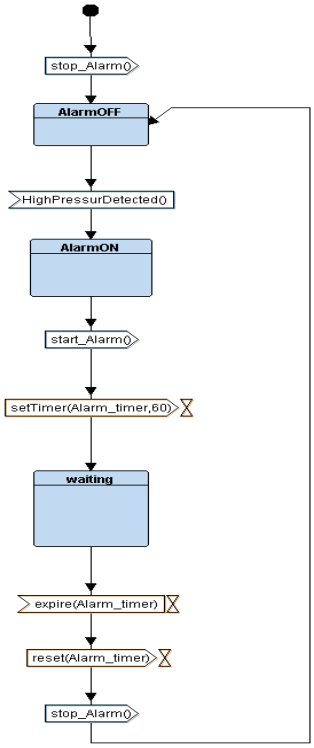
main algorithms:

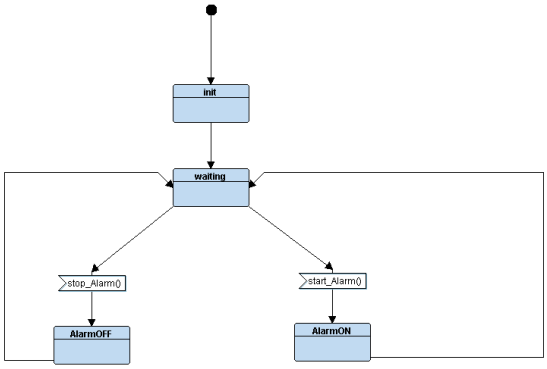


PRESSURE SENSOR DRIVER



ALARM MONITOR AND ACTUATOR





CODING

to implement the system hereafter, we present some shots obj files also we can see our own Startup.c the linker script and the makefile to automate the building. First, we created headers (look on GitHub repository)

STARTUP.C

```
#include<stdint.h>

extern uint32_t _E_text;
extern uint32_t _S_data;
extern uint32_t _E_data;
extern uint32_t _S_bss;
extern uint32_t _E_bss;


extern void main();

void reset_handler()
{
       uint32_t data_size = (unsigned char*)&_E_data - (unsigned char*)& _S_data;
   unsigned char * p_scr= (unsigned char*)& _E_text;
       unsigned char * p_dst= (unsigned char*)& _S_data;
    int i;
    for(i=0;i<data_size;i++)
    {
       *(unsigned char*) p_dst++ = * (unsigned char*) p_scr++;
    }
```

```
    uint32_t bss_size = (unsigned char*)&_E_bss - (unsigned char*)& _S_bss;
    p_dst= (unsigned char*)& _S_bss;

    for(i=0;i<bss_size;i++)
    {
       *(unsigned char*) p_dst++ = (unsigned char) 0;
    }

       main();
}

void default_handler()
{
       reset_handler();
}
void nmi_handler() __attribute__((weak,alias("default_handler")));
void h_fault_handler() __attribute__((weak,alias("default_handler")));
void bus_fault() __attribute__((weak,alias("default_handler")));


static unsigned long stack_top [256];
void (* const g_vector [])() __attribute__((section(".vectors"))) =
   {
       (void(*)()) (unsigned long) &stack_top [256],
       &reset_handler,
          &nmi_handler,
      &h_fault_handler,
      &bus_fault
   };
```

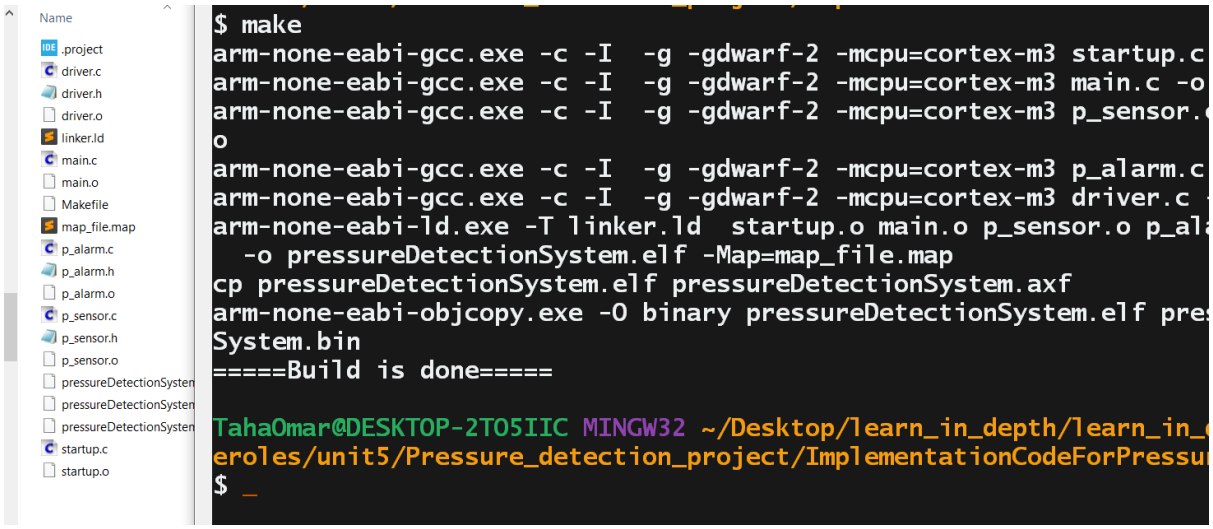
LINKER SCRIPT

```
MEMORY
{
    flash(rx) : o = 0X00000000, l = 512M
    sram(rwx):  o = 0X20000000, l = 512M
}

SECTIONS
{
    .text : {
      *(.vectors*)
      *(.text*)
      _E_text = .;
    }>flash

    .data :{
      _S_data = .;
      *(.data*)
      _E_data = .;
    }>sram AT> flash

    .bss :{
      _S_bss = . ;
      *(.bss*)
      _E_bss = .;
      _stack_top = . +0x1000;
    }>sram
}
```

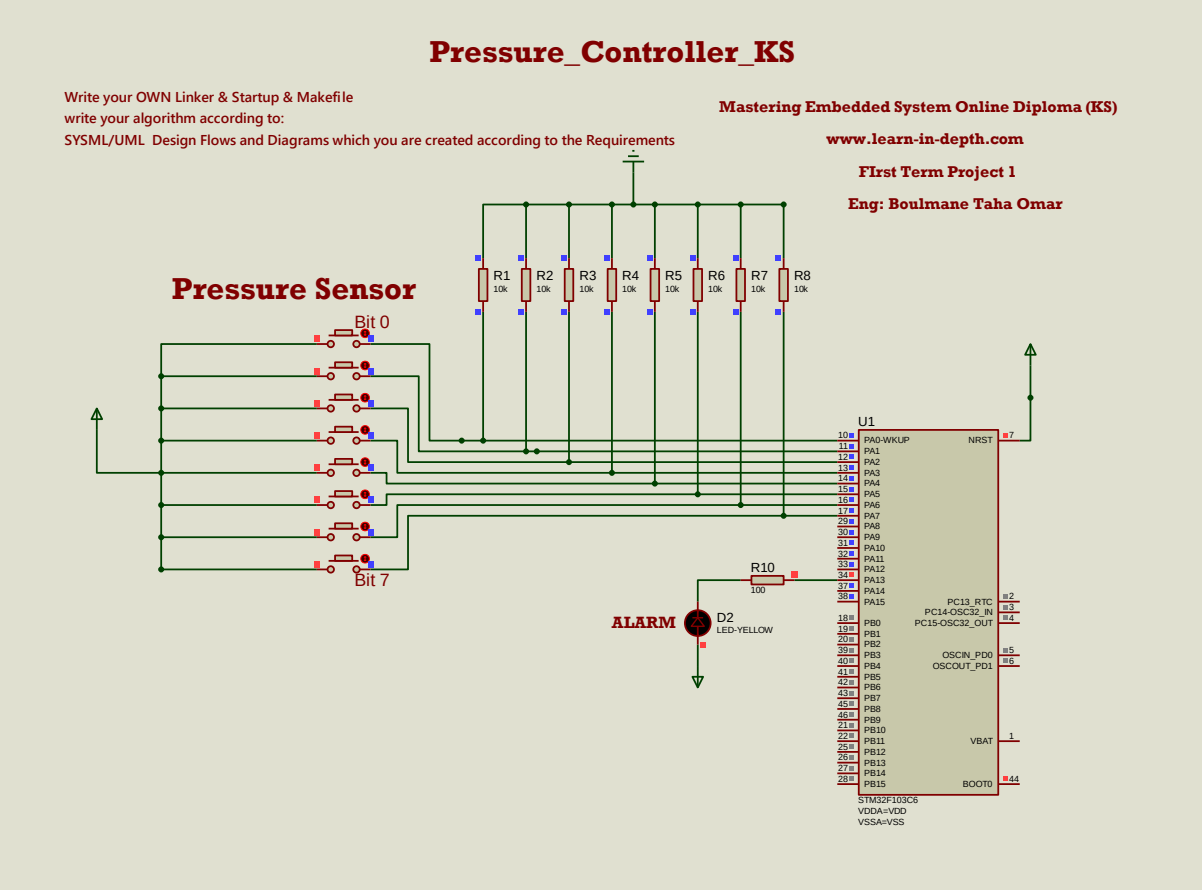
MAKEFILE

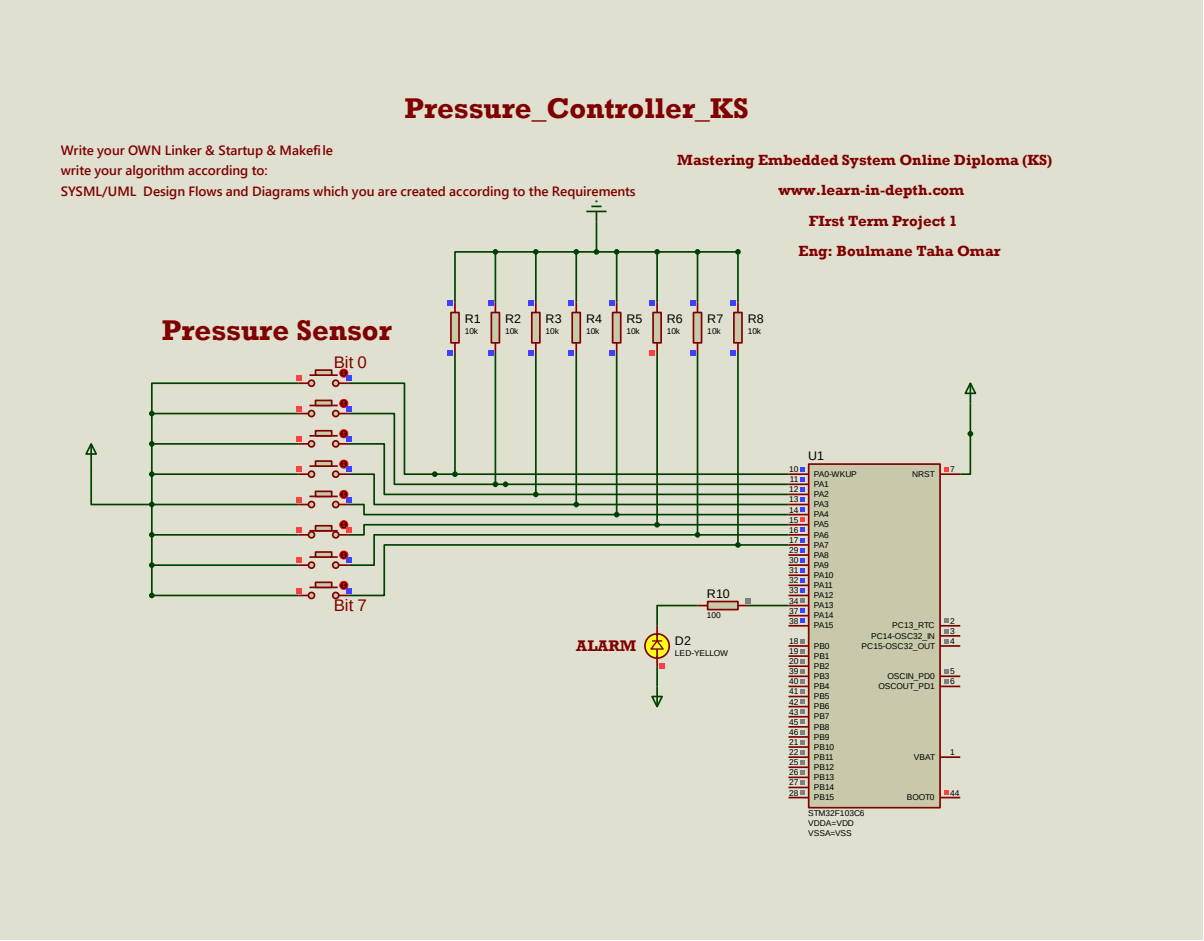
```
#First term project makefile by Eng: Taha Omar Boulmane
CC=arm-none-eabi-
CFLAGS= -g -gdwarf-2 -mcpu=cortex-m3

INCS=-I
LIBS=
SCR =$(wildcard *.c)
OBJ =$(SCR:.c=.o)
AS  =$(wildcard *.s)
ASOBJ=$(AS:.s=.o)
Project_name=pressureDetectionSystem

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

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

$(Project_name).elf: $(OBJ) $(ASOBJ)
	$(CC)ld.exe -T linker.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 $< $@

symbol:$(Project_name).elf
	$(CC)nm.exe $<

header:$(Project_name).elf
	$(CC)objdump.exe -h $<

asm:$(Project_name).elf
	$(CC)objdump.exe -d $<
clean:
	rm *.elf *.bin
clean_all:
	rm *.elf *.bin *.o
```

MAKE RESULTS



TEST AND SIMULATION





(In this report I used data from other engineers repositories for that I thank them all)