Mastering Embedded System Online Diploma
www.learn-in-depth.com

First Term (Final Project 1)

High Pressure Detection

Eng. Salma Sherif

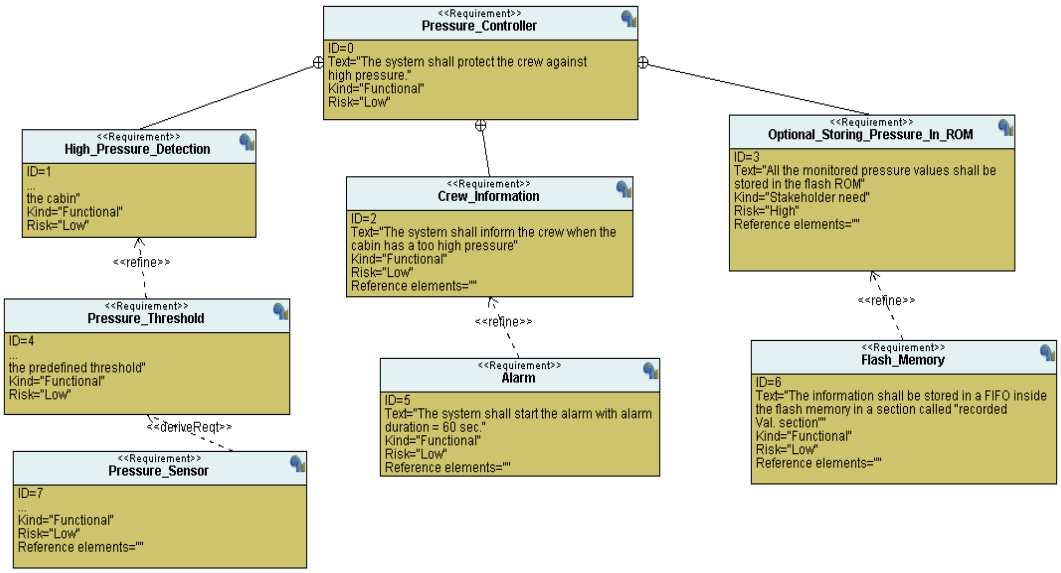
Overview:

This is a report about the Pressure Controller System in an airplane which informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin. The alarm duration equals 60 seconds. This project is done from scratch, with a written linker script, a startup and a make file.

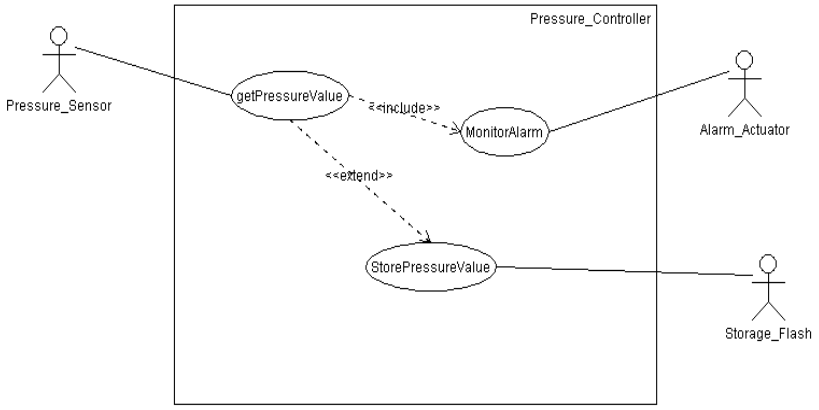
The report will include UML diagrams, screenshot from the simulation demo and the code.

STM32F106C6 microcontroller is used in the implementation.

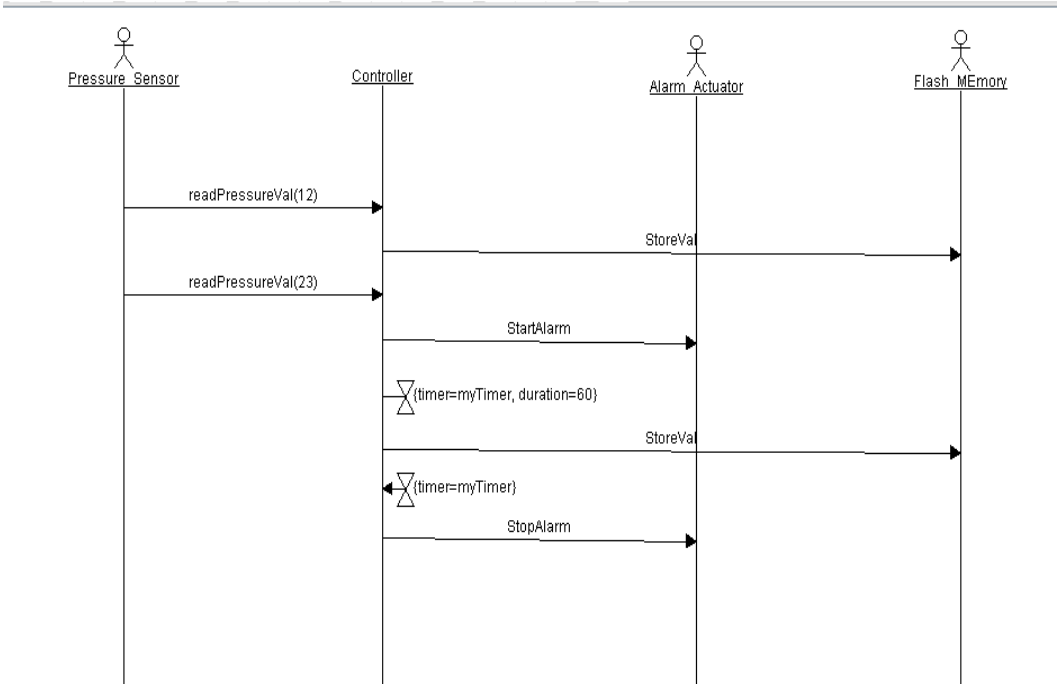
1. Requirement Diagram



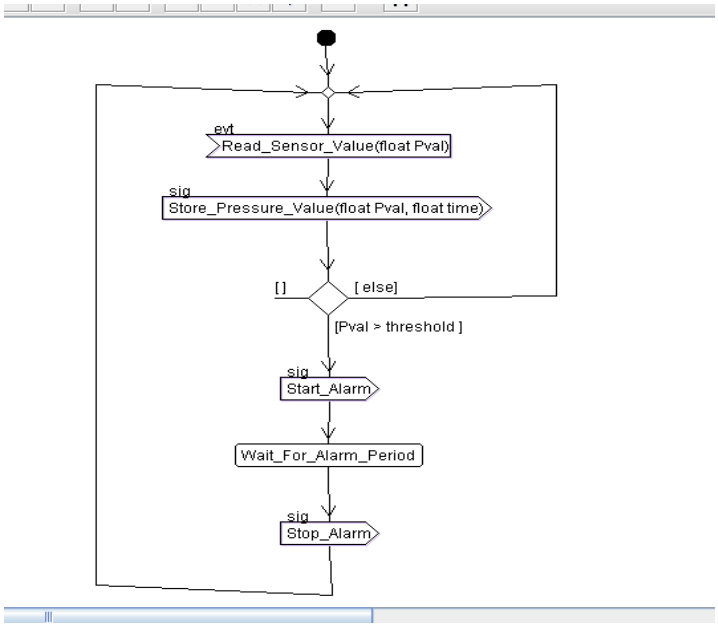
2. Use Case Diagram



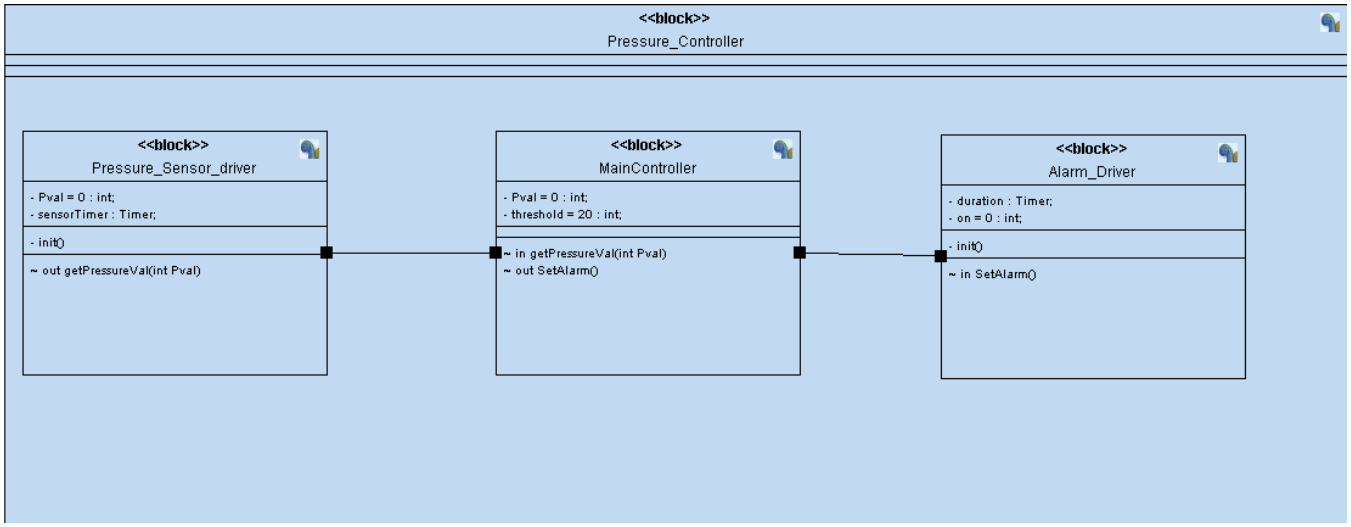
3. Sequence Diagram



4. Activity Diagram

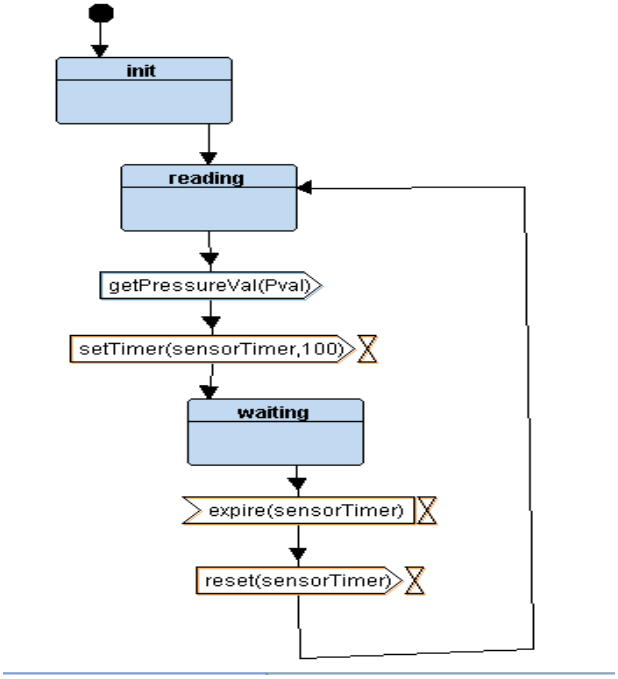


5. System Design



6. State Machines with its corresponding .c & .h files

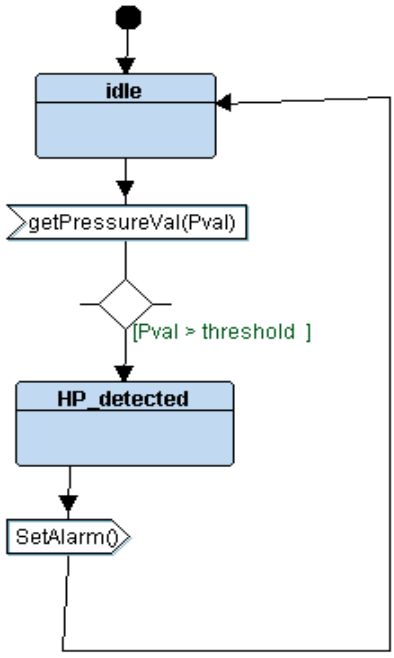
6.1. Pressure Sensor Driver




```c
/*...
#include "PressureSensorDriver.h"
int getPressureVal();
//int generate_random(int l, int r, int c);

int pVal = 0;
void (*PS_state)();

void PS_init(){
    PS_state = STATE(PS_reading);
    //printf("INIT PS \n");
}

int getPressureVal(){
    //return generate_random(15,25,1);
    return (GPIOA_IDR & 0xFF);
}

STATE_define(PS_waiting){
    //printf("Sensor state : Waiting......\n");
    Delay(100000);
    PS_state = STATE(PS_reading);
}

STATE_define(PS_reading){
    PS_states_id = PS_reading;
//  printf("Sensor state : reading.....\n");
    pVal = getPressureVal();
    get_pressure_value(pVal);
    PS_state = STATE(PS_waiting);
}
```

```c
/*
 * PressureSensor.h
 *
 *  Created on: Sep 5, 2021
 *      Author: Salma Sherif
 */

#ifndef PRESSURESENSORDRIVER_H_
#define PRESSURESENSORDRIVER_H_

#include "states.h"
#include "driver.h"
//#include "stdlib.h"

enum{
    PS_reading,
    PS_waiting
}PS_states_id;


void PS_init();
STATE_define(PS_reading);

extern void (*PS_state)();
#endif /* PRESSURESENSORDRIVER_H_ */

```

6.2. Main Controller



MainController.c | AlarmDriver.c | PressureSensorDriver.c

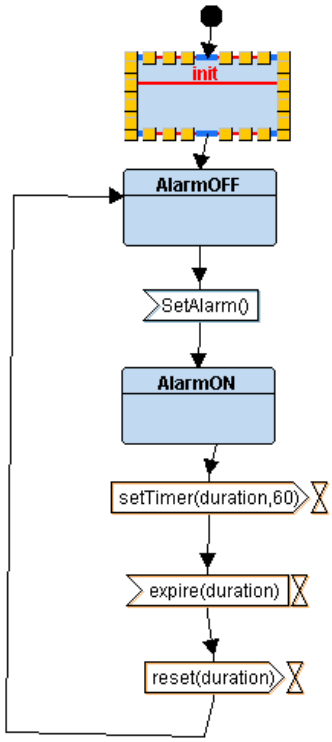
D: > embedded diploma > REPO > embedded-systems-diploma > First_Term_Project1

```c
/*...
#include "MainController.h"

int threshold = 20;

void (*C_state)() = STATE(idle);


void get_pressure_value(int P){
   // printf("Sensor reads :  %d\n", P);
    if (P > 20){
        C_state = STATE(HP_detected);

    }else{
        C_state = STATE(idle);
    }
}

STATE_define(HP_detected){
    C_id = HP_detected;
   // printf("Controller in HP detected state .....\n");
    Set_Alarm_actuator(1);
    C_state = STATE(idle);

}

STATE_define(idle){
    Set_Alarm_actuator(0);
    C_id = idle;
   // printf("Controller in idle state .....\n");

}
```

D: > embedded diploma > REPO > embedded-systems

```c
/*
 * MainController.h
 *
 *  Created on: Sep 5, 2021
 *      Author: Salma Sherif
 */

#ifndef MAINCONTROLLER_H_
#define MAINCONTROLLER_H_

#include "driver.h"
#include "states.h"
#include "PressureSensorDriver.h"


enum{
    idle,
    HP_detected

}C_id;

STATE_define(HP_detected);
STATE_define(idle);


extern void (*C_state)();

#endif /* MAINCONTROLLER_H_ */

```

6.3. Alarm Driver

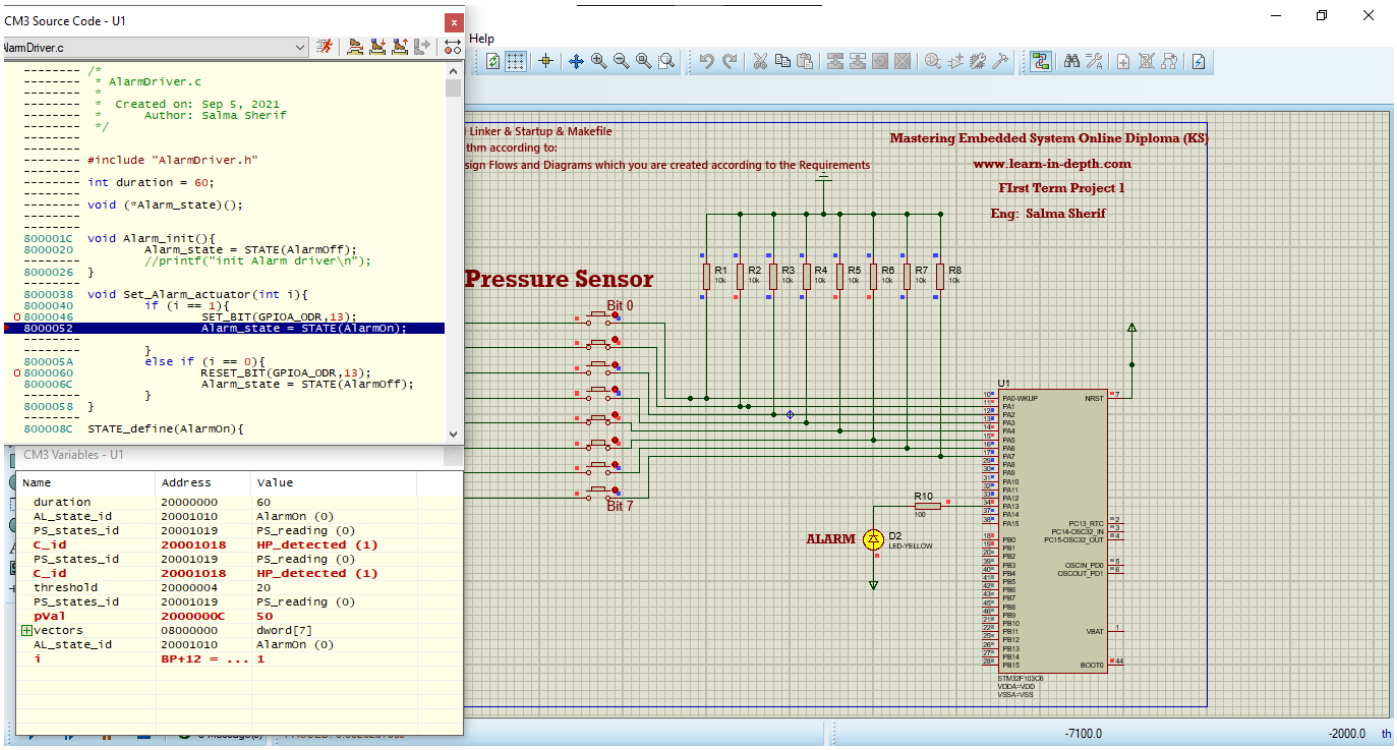



```c
/*...
#include "AlarmDriver.h"
int duration = 60;
void (*Alarm_state)();

void Alarm_init(){
    Alarm_state = STATE(AlarmOff);
    //printf("init Alarm driver\n");
}

void Set_Alarm_actuator(int i){
    if (i == 1){
        SET_BIT(GPIOA_ODR,13);
        Alarm_state = STATE(AlarmOn);
    }else if (i == 0){
        RESET_BIT(GPIOA_ODR,13);
        Alarm_state = STATE(AlarmOff);
    }
}

STATE_define(AlarmOn){
//  printf("Alarm is ONNNNNN!!\n");
    AL_state_id = AlarmOn;
//  printf("Delaaaaay!!!\n");
    Delay(duration*1000); //delay 60 sec
    Alarm_state = STATE(AlarmOff);
}

STATE_define(AlarmOff){
    //printf("Alarm is OFFF!!\n");
    AL_state_id = AlarmOff;
}
```

```c
/*
 * AlarmDriver.h
 *
 *  Created on: Sep 5, 2021
 *      Author: Salma Sherif
 */

#ifndef ALARMDRIVER_H_
#define ALARMDRIVER_H_

#include "driver.h"
#include "states.h"



enum{
    AlarmOn,
    AlarmOff

}AL_state_id;

void Alarm_init();
STATE_define(AlarmOn);
STATE_define(AlarmOff);


extern void (*Alarm_state)();

#endif /* ALARMDRIVER_H_ */

```

7. Simulation Results

The measure pressure was 50 which is greater than 20, so the alarm is ON for 60 seconds



After the 60 seconds ended, the Alarm is turned Off, and would turn on again if the pressure is greater than 20.