Mastering Embedded System Online Diploma

www.learn-in-depth.com

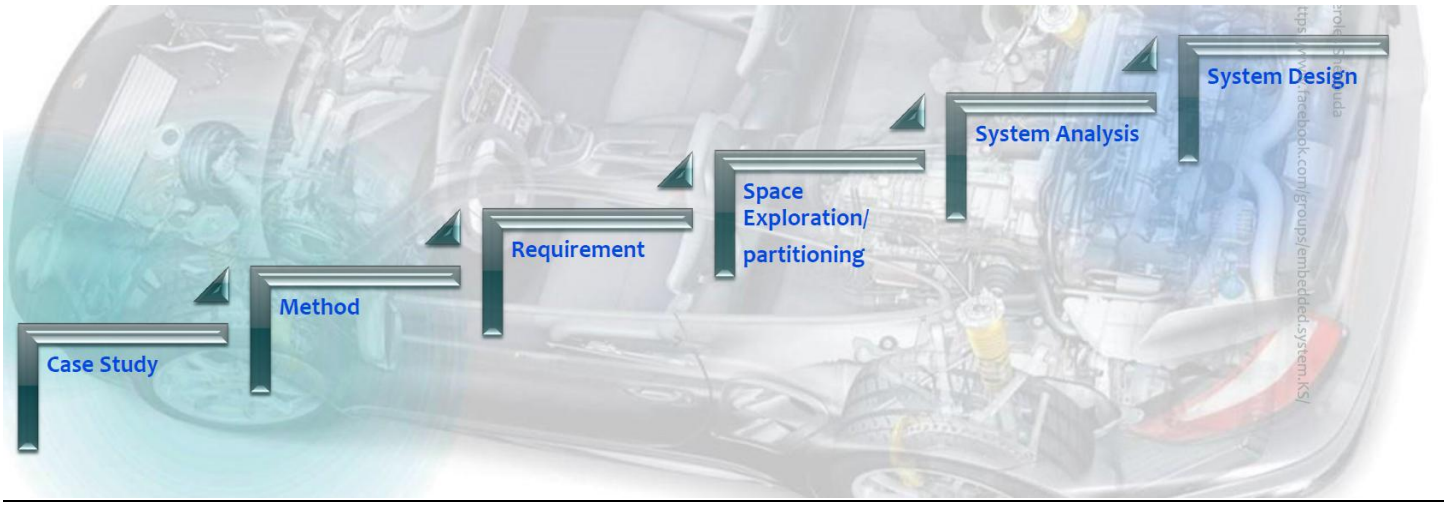
High Pressure Detection System

First Term (Final Project 1)

My name: Eng. Mina Gamil Gaeed

My profile: minagamil.ga@gmail.com

High Pressure Detection System



---

1. Case Study:

- We need to make a program that Indicate the airplane crew when the pressure in the cabin is higher than certain value

1.1. Specifications: (From Client)

1.1.1. Pressure in the cabin should not exceed 20 bars.
1.1.2. Indication using alarm and Red LED light for pressure higher than 20 bar.
1.1.3. Indication using Green LED light if pressure less than 20 bars.
1.1.4. Indication using Yellow LED light if pressure is equal to 20 bars.
1.1.5. Alarm Duration is 60 Seconds.
1.1.6. Keep track of the measured values and store them in Flash memory.

1.2. Assumptions:

1.2.1. Controller startup and shutdown are not modeled.
1.2.2. Controller maintenance is not modeled.
1.2.3. Pressure sensor never fails.
1.2.4. Alarm never fails.
1.2.5. LED Light never fails.
1.2.6. Controller never faces power cut.

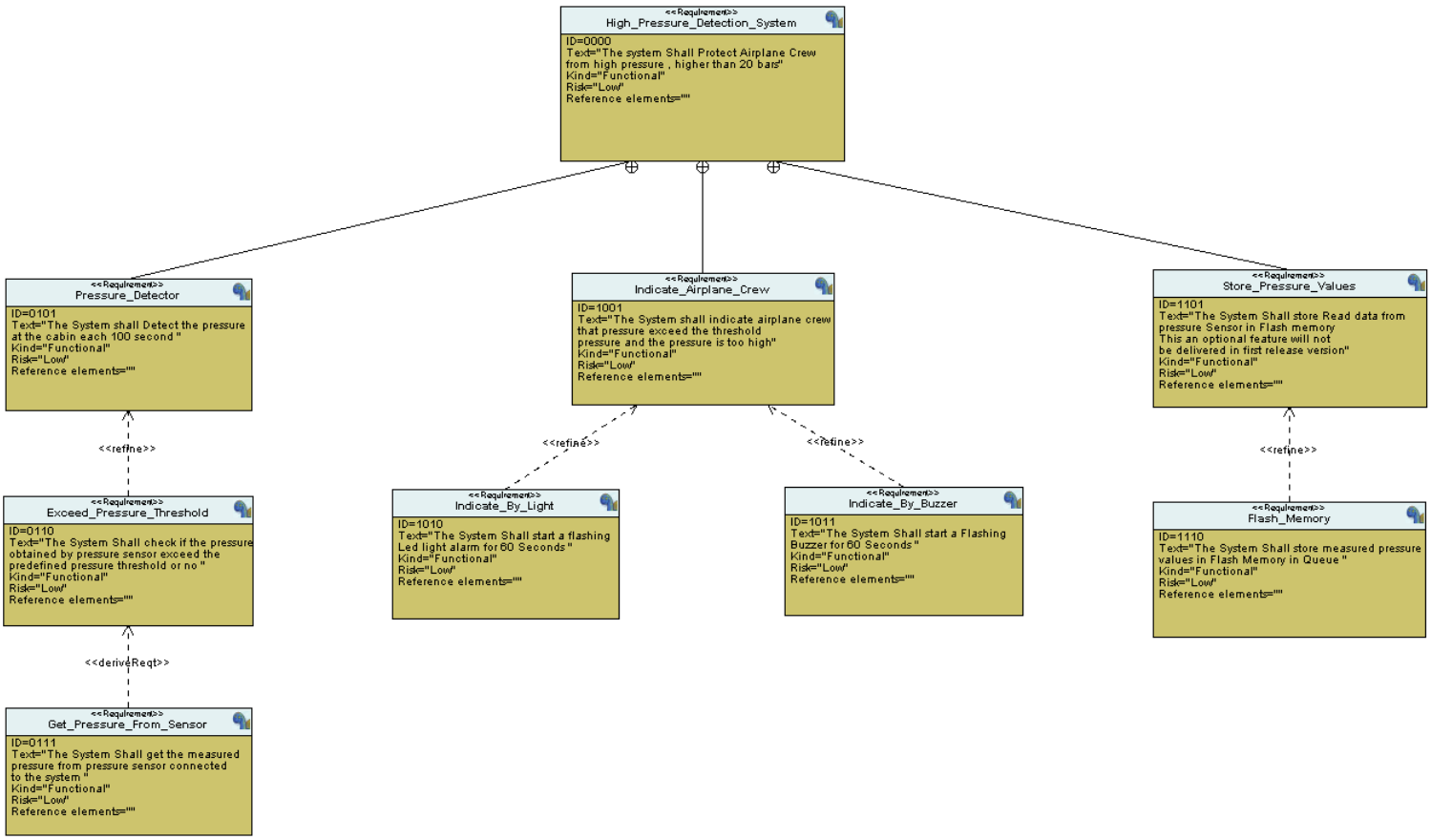
1.3. Versioning:

Keep track of measured values option is not modeled in first version of design.

2. Method:

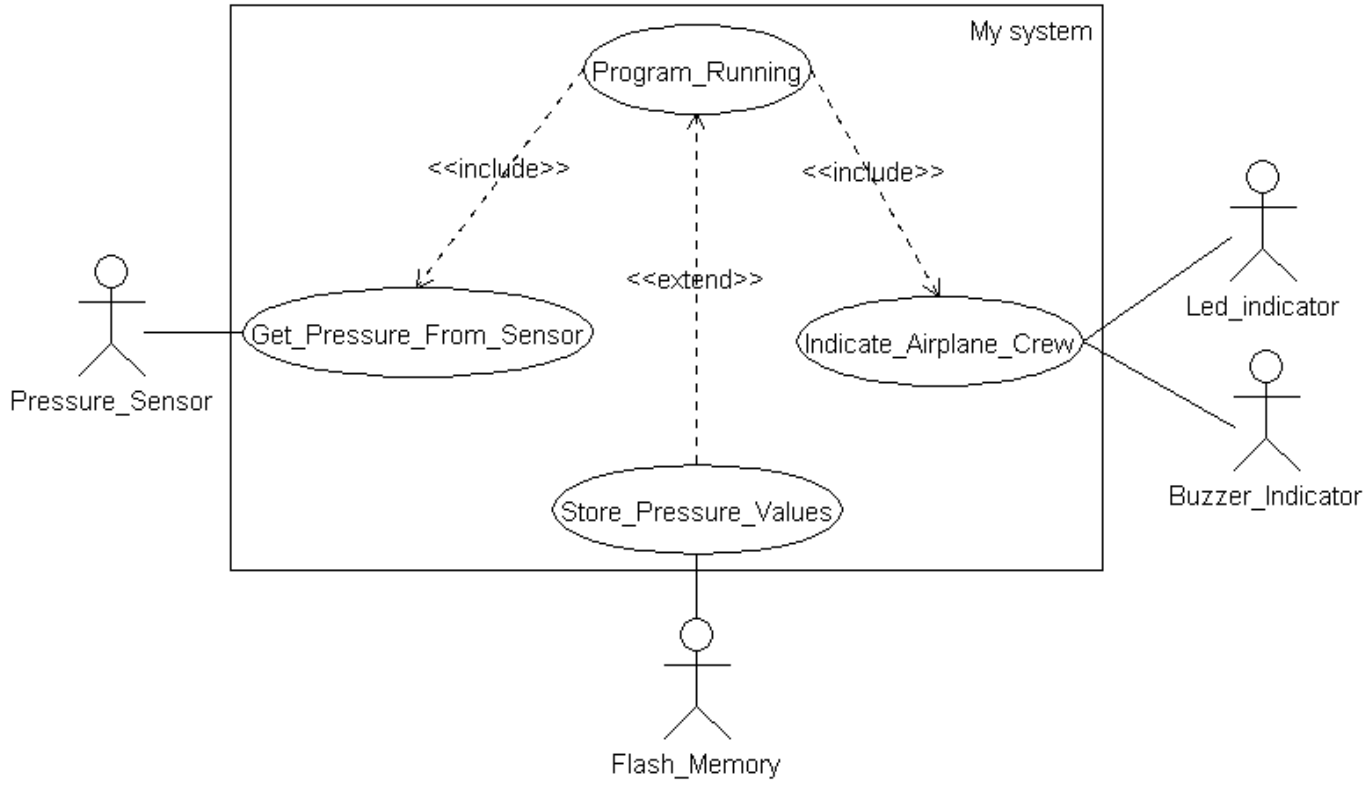
We will use Waterfall model -SDLC

3. Requirements:

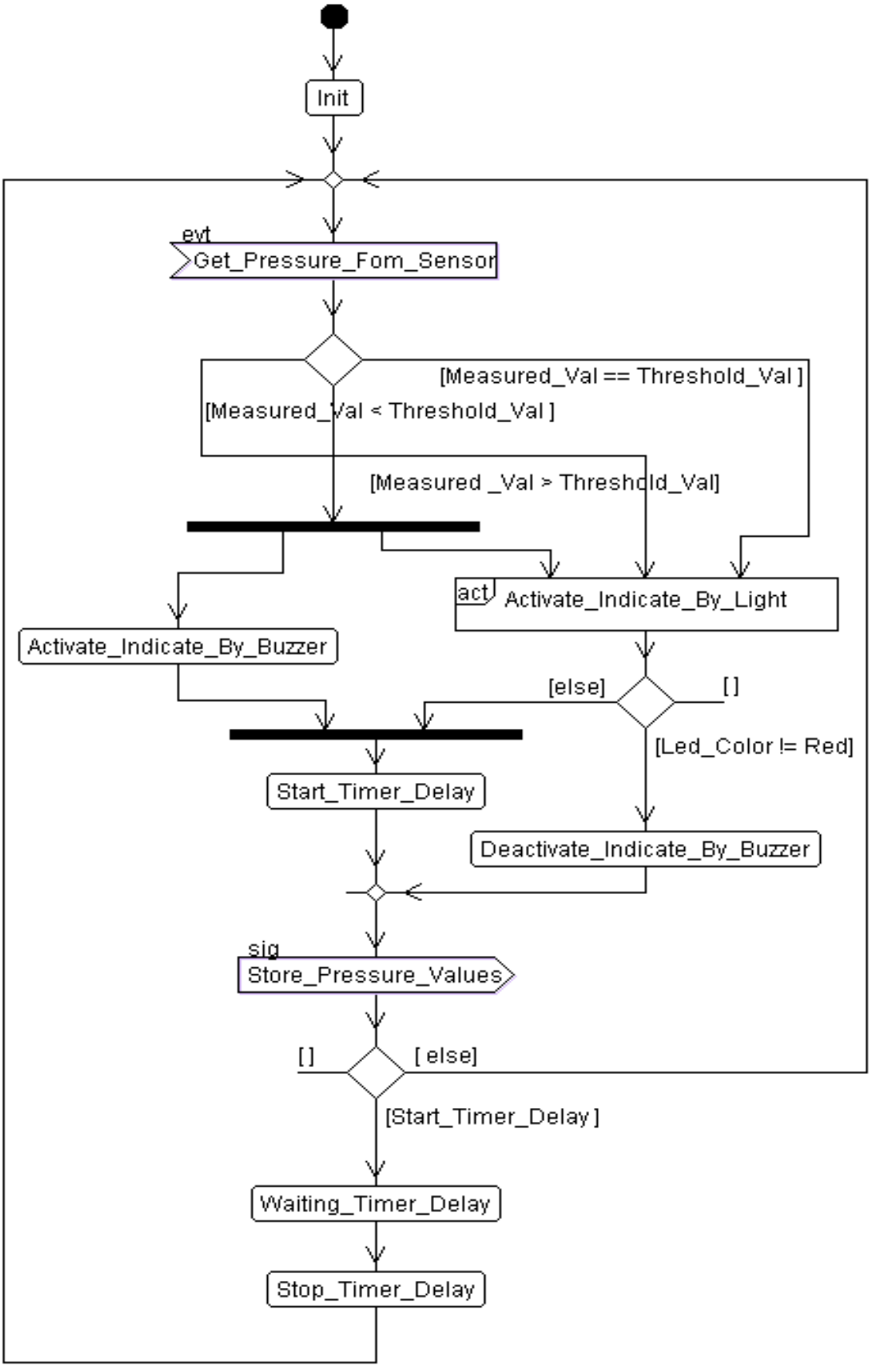


4. System Analysis:

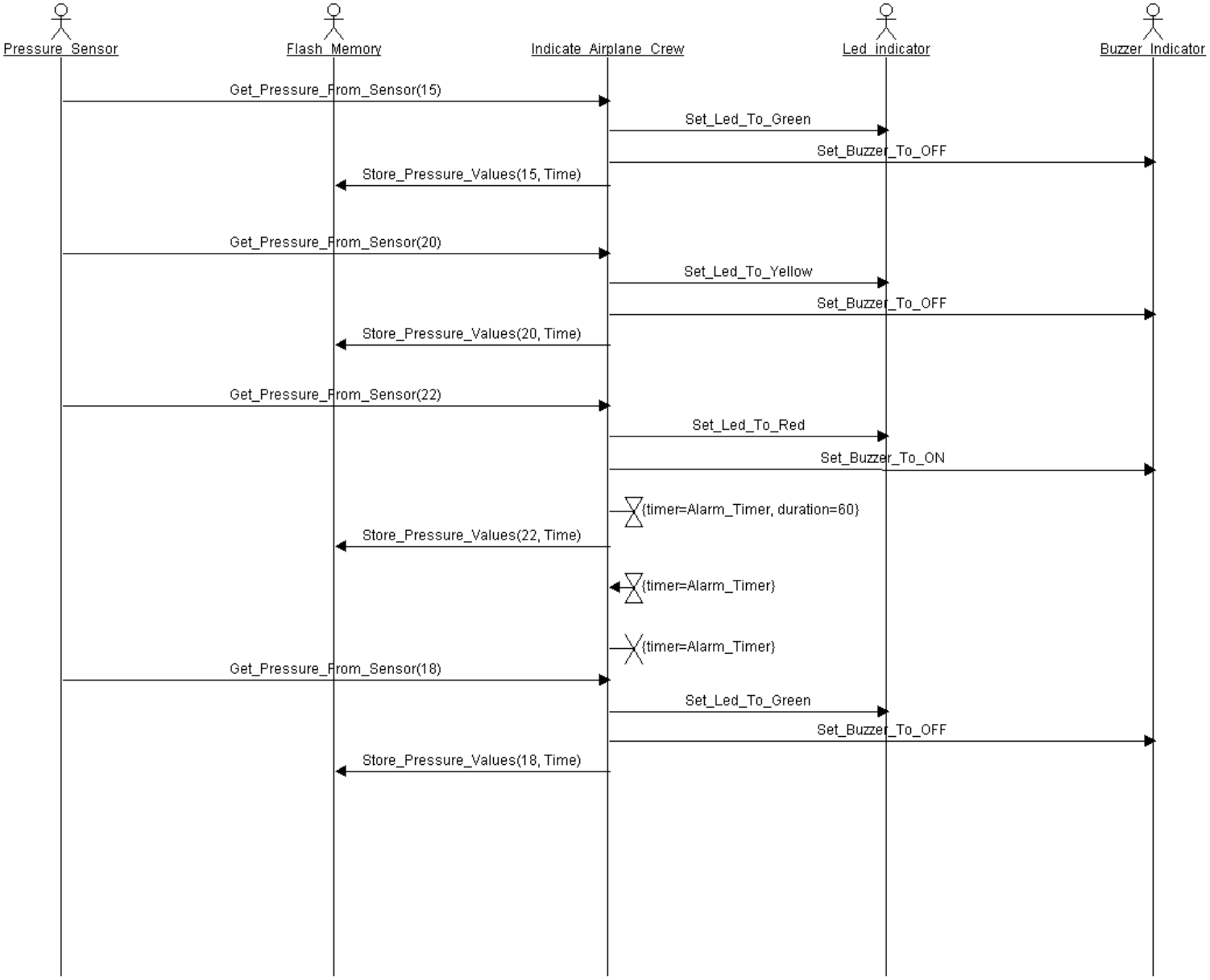
4.1. Use Case Diagram:



4.2. Activity Diagram:

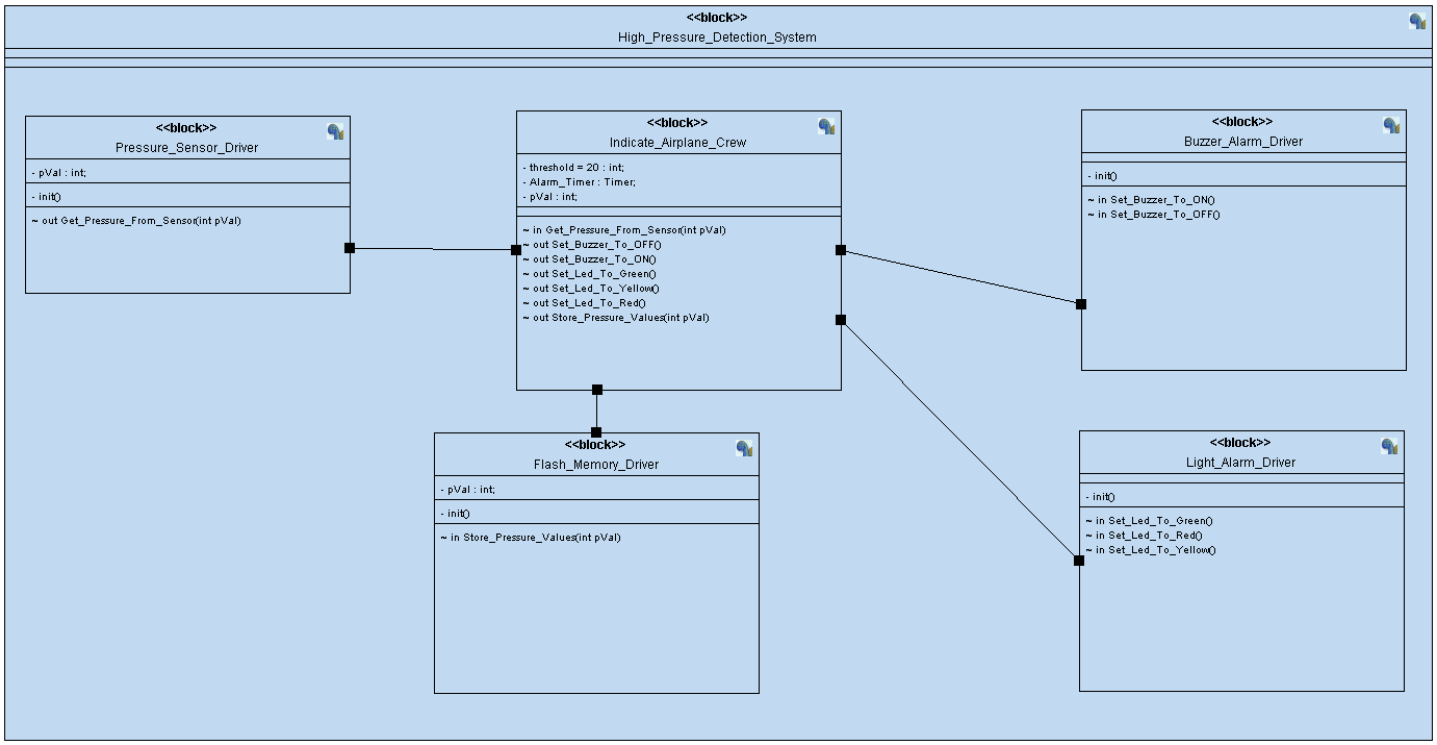


4.3. Sequence Diagram:



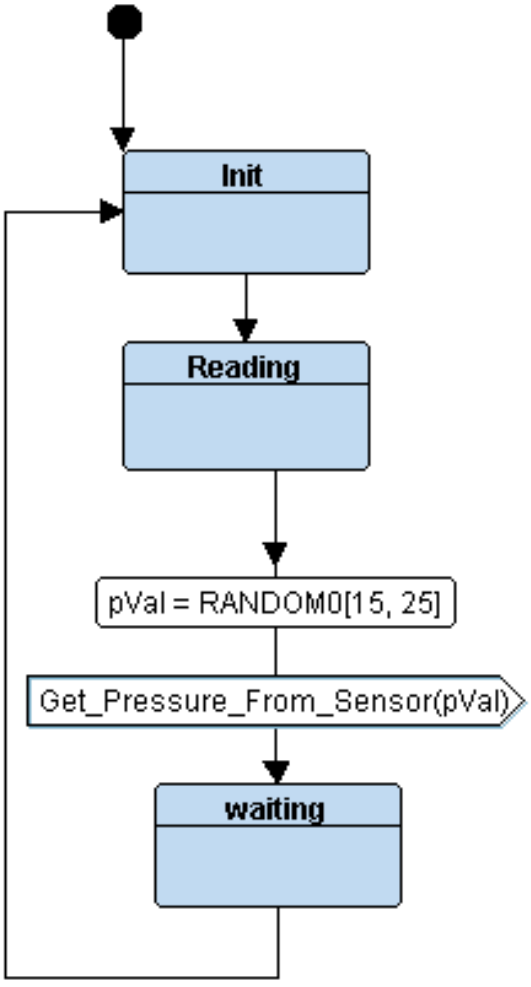
5. System Design:

5.1. Block Diagram:

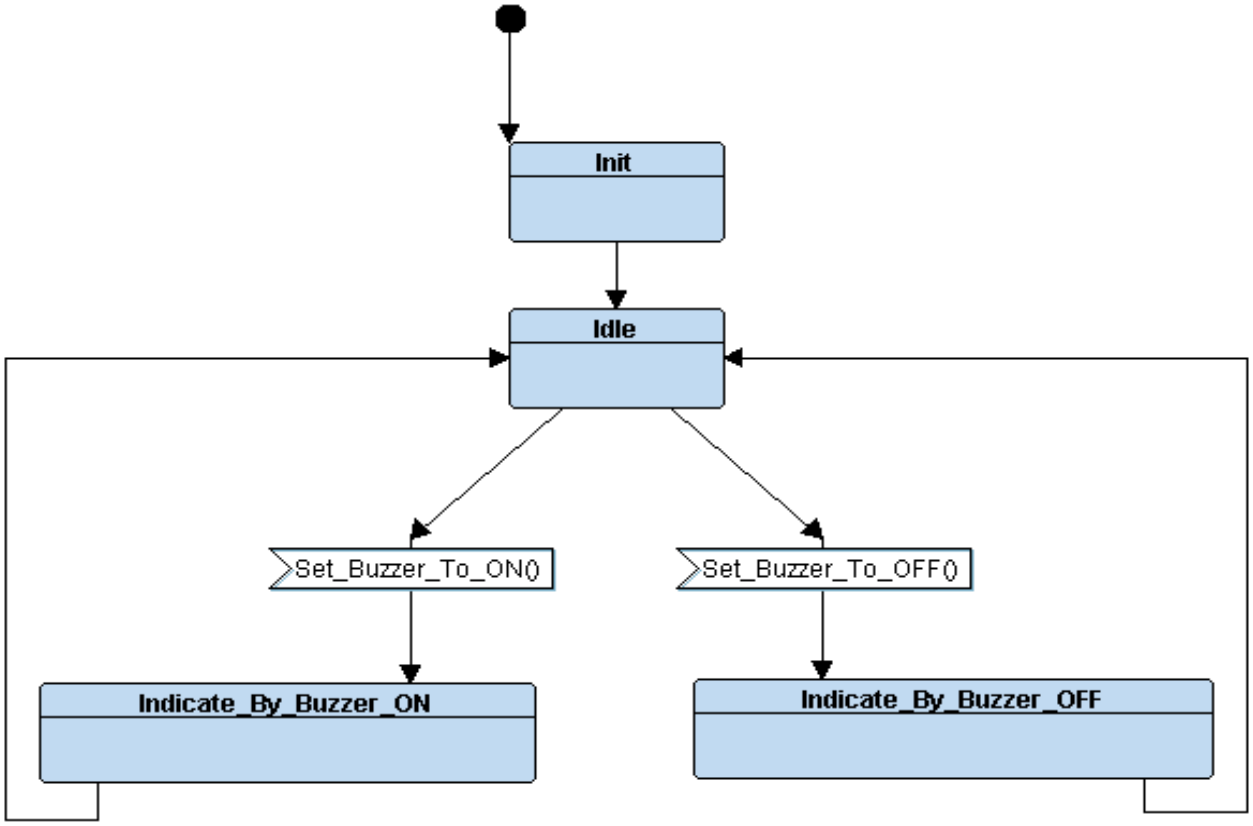


5.2. State Machine:

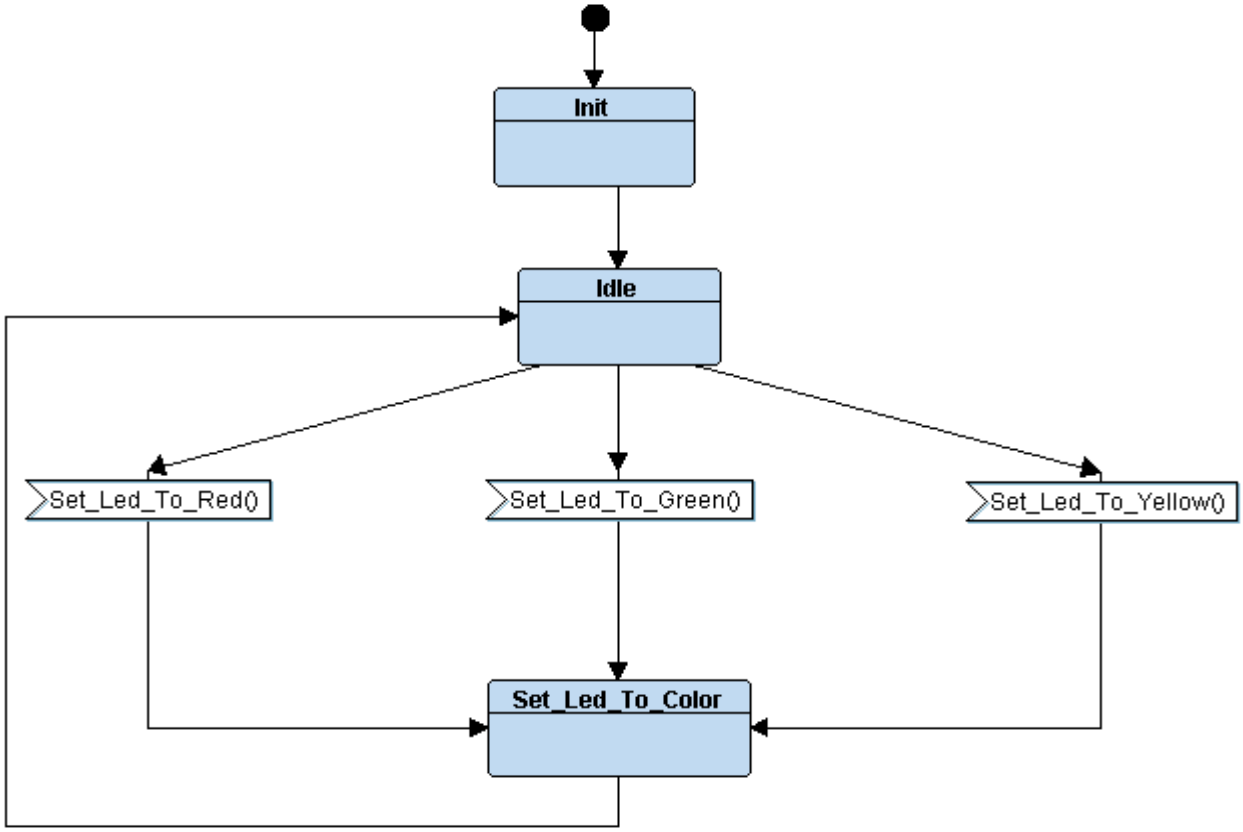
5.2.1. Pressure Sensor.



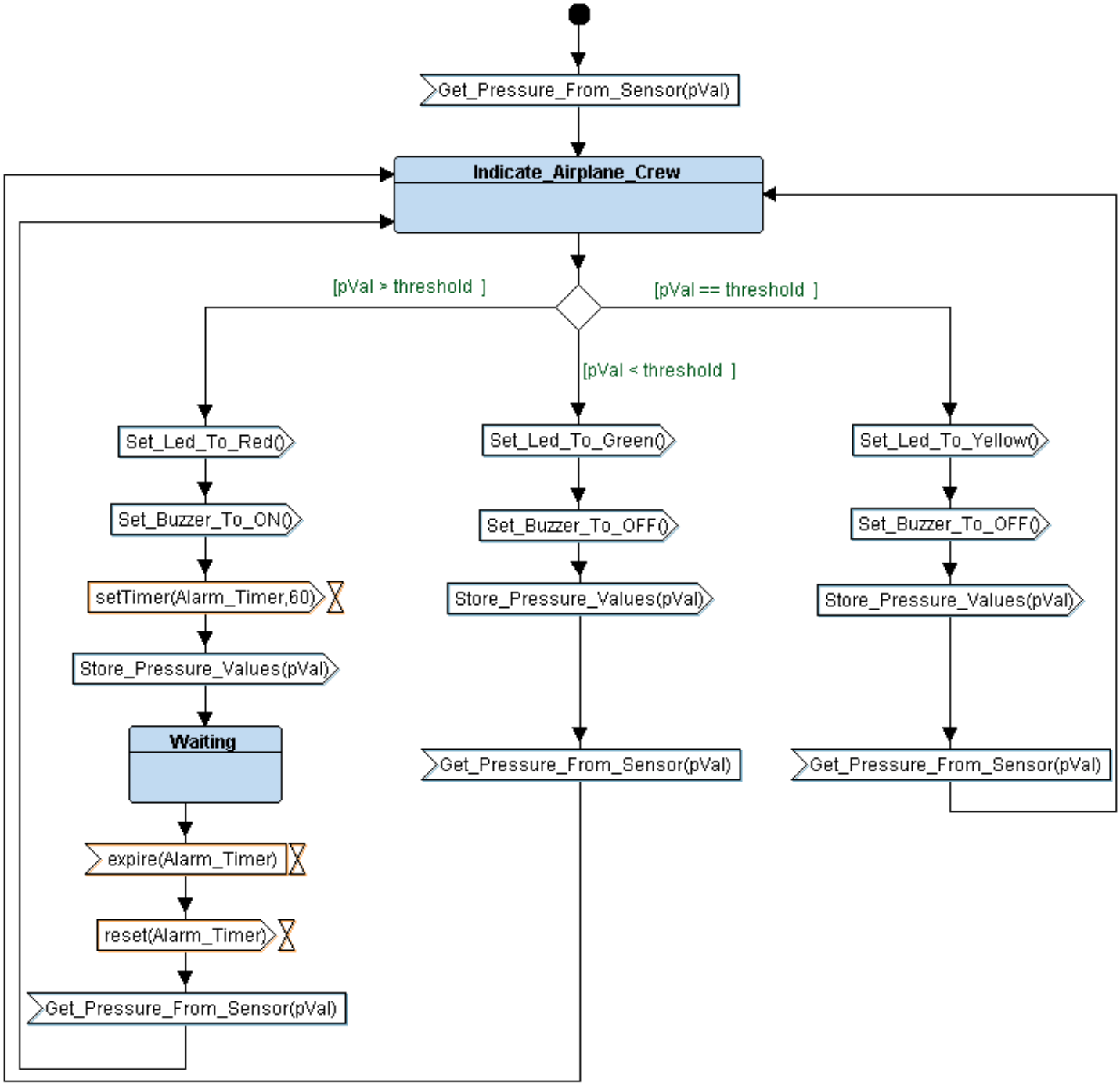
5.2.2. Buzzer Alarm Driver



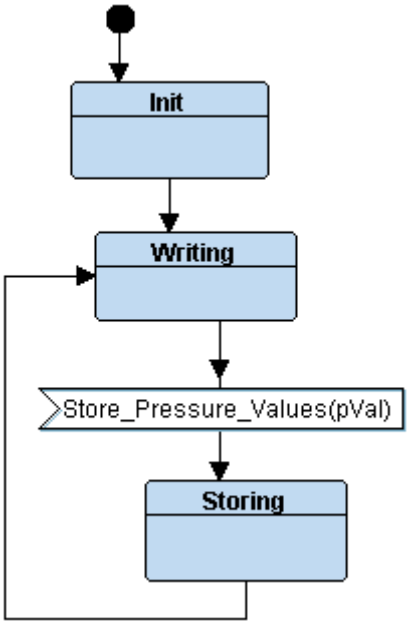
5.2.3. Light Alarm Driver.



5.2.4. Indicate Airplane Crew.

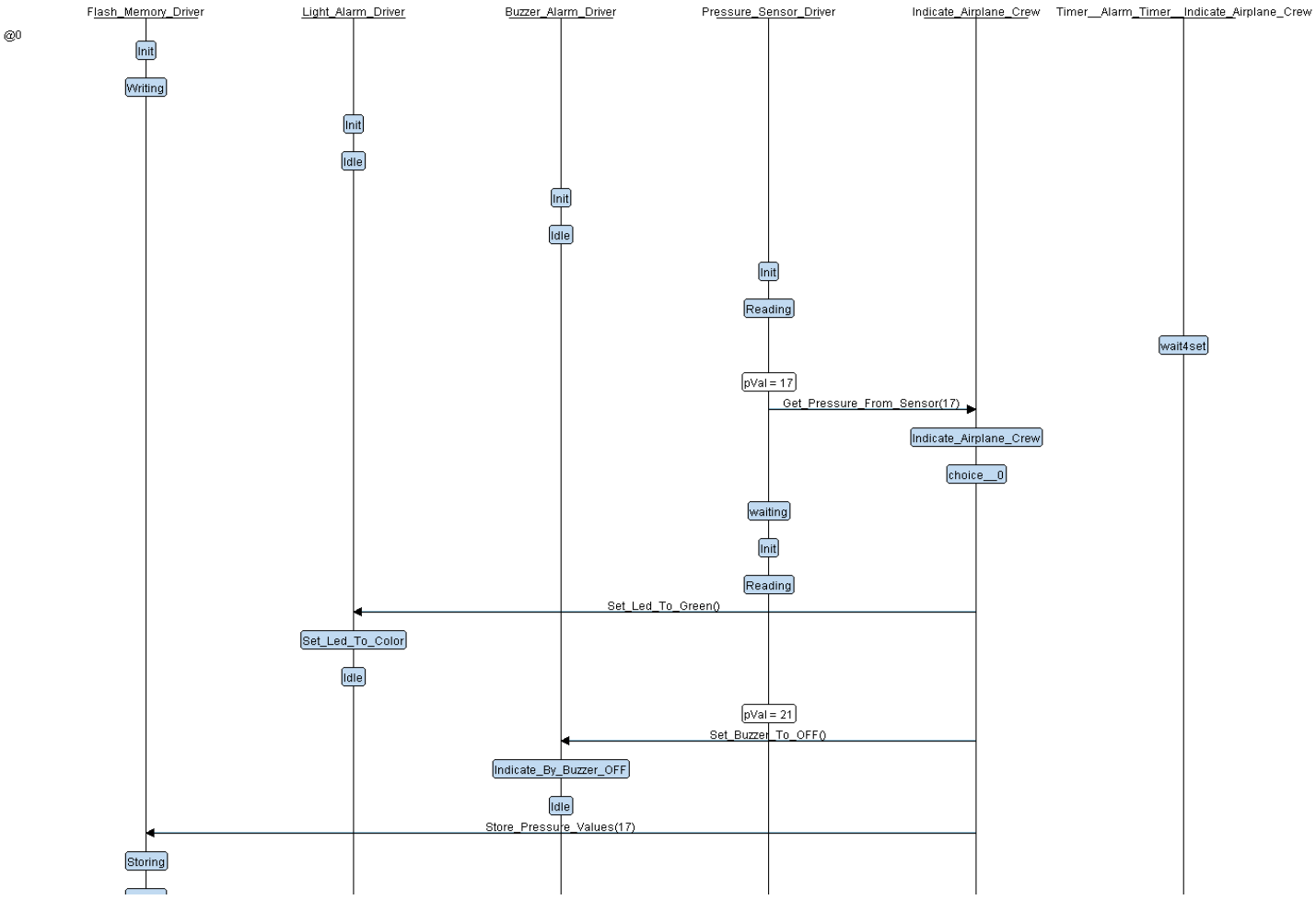


5.2.5. Flash Memory Driver.

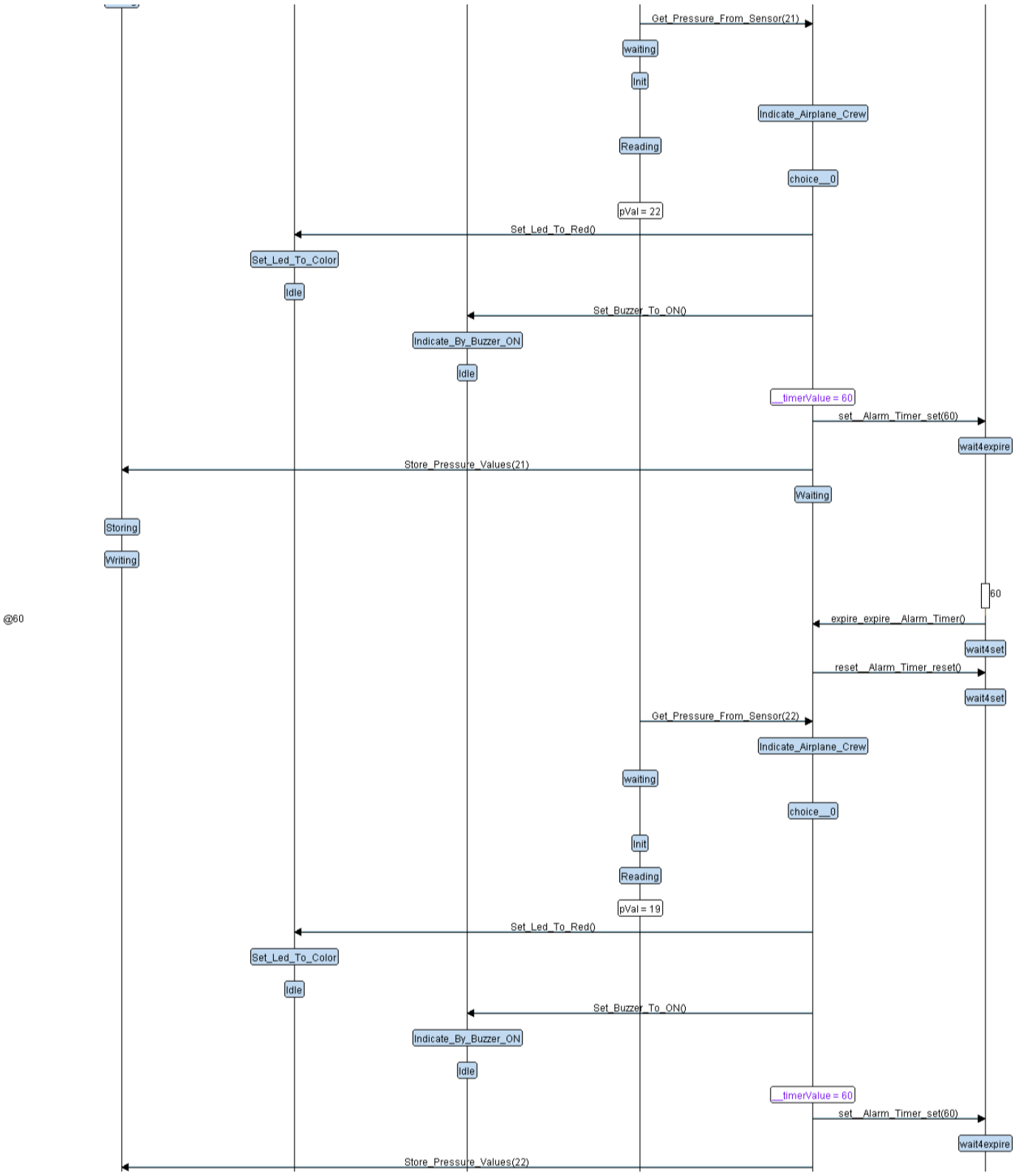


5.3. Trace Sequence Diagram.

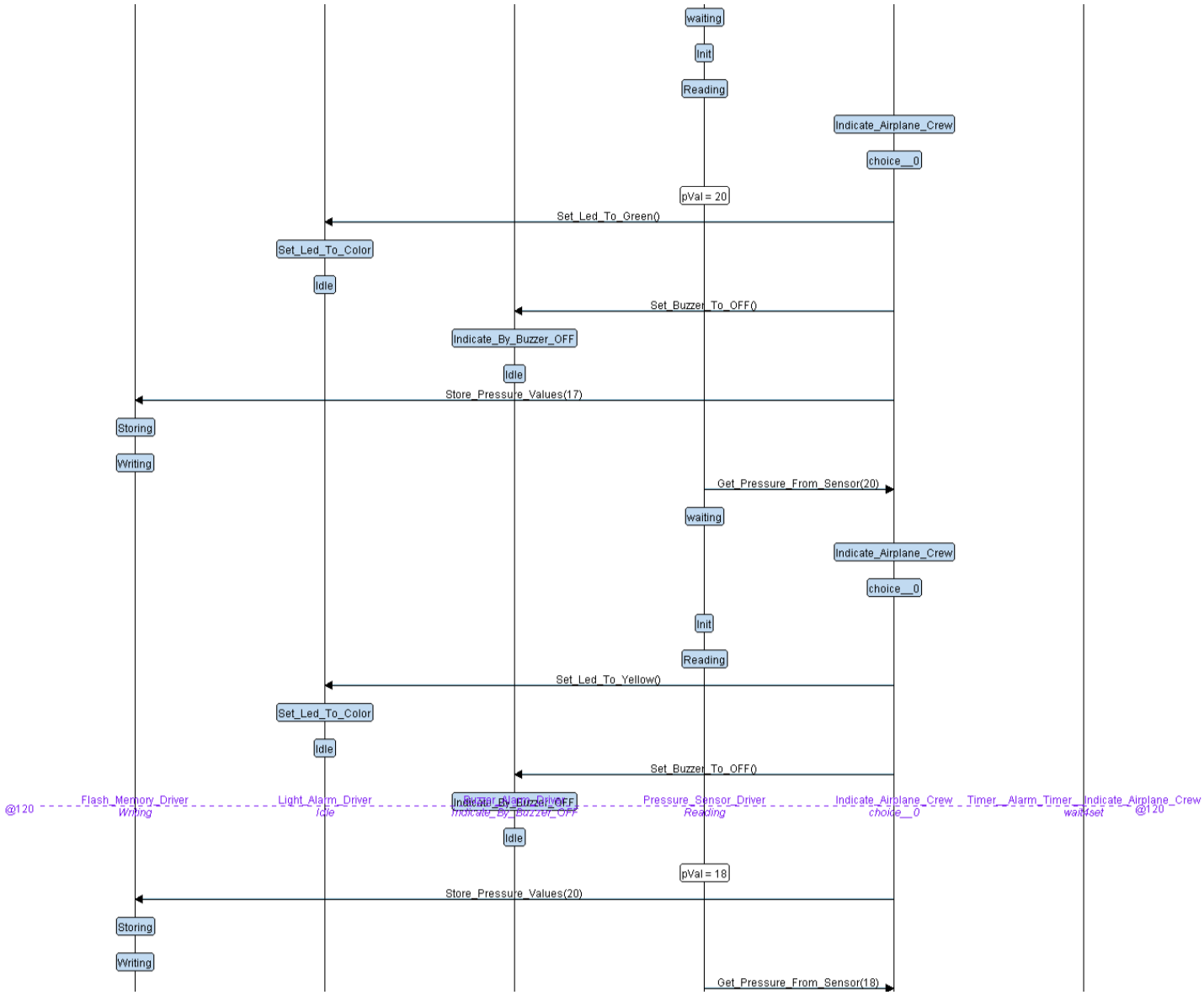
5.3.1. Case 1: Pressure Value < 20 bars



5.3.2. Case 2: Pressure Value > 20 bars



5.3.3. Case 3: Pressure Value == 20 bars



6. Code Implementation:

6.1. Pressure Sensor:

6.1.1. Pressure_Sensor_Driver.c

```c
/**
 ************************************************************
 *@file      : Pressure_Sensor_Driver.c
 *@author    : Mina Gamil
 *@date      : Feb 04, 2025
 *@brief     : C program for Get pressure From Sensor States
 ************************************************************
 */

/**** Include Module header ****/
#include "Pressure_Sensor_Driver.h"

/**** Declare and assign used variable ****/
vuint32 pVal;

/*** Pressure_Val_State_Id enum ***/
enum{
    Reading_Pressure_Val,
    Waiting_Pressure_Val
}Pressure_Val_State_Id;

/** Pressure_Sensor pointer to function **/
void (*Pressure_Sensor_ptr2Func)();

/* Pressure_Sensor Init API*/
void Pressure_Sensor_Init()
{
    Pressure_Sensor_ptr2Func = STATE(Waiting_Pressure_Val);          // Set Pressure_Sensor pointer to function equal to the address of waiting state
}

/* Reading Pressure Value State*/
STATE_Define(Reading_Pressure_Val)
{
    Pressure_Val_State_Id = Reading_Pressure_Val;                    // set Pressure_Val_State_Id = Reading_Pressure_Val
    pVal = getPressureVal();                                        // call getPressureVal function to get the pressure value and store it in pVal
    Pressure_Sensor_ptr2Func = STATE(Waiting_Pressure_Val);          // Then set Pressure_Sensor Pointer to waiting state
}

/* Waiting Pressure Value State*/
STATE_Define(Waiting_Pressure_Val)
{
    Pressure_Val_State_Id = Waiting_Pressure_Val;                    // set Pressure_Val_State_Id = Waiting_Pressure_Val
    Pressure_Sensor_ptr2Func = STATE(Reading_Pressure_Val);          // Set Pressure_Sensor pointer to Reading state
    Pressure_Sensor_ptr2Func();                                     // Run pointed Function to get new pressure value
}
```

6.1.2. Pressure_Sensor_Driver.h

```c
/*
 * Pressure_Sensor_Driver.h
 *
 *  Created on: Feb 04, 2025
 *      Author: Mina Gamil
 */

#ifndef PRESSURE_SENSOR_DRIVER_H_
#define PRESSURE_SENSOR_DRIVER_H_

/**** Include state header ****/
#include "state.h"

/** Pressure_Sensor pointer to function **/
extern void (*Pressure_Sensor_ptr2Func)();

/* Pressure_Sensor_Driver API's */
void Pressure_Sensor_Init();
STATE_Define(Reading_Pressure_Val);
STATE_Define(Waiting_Pressure_Val);

#endif /* PRESSURE_SENSOR_DRIVER_H_ */
```

6.2. Buzzer Alarm:

6.2.1. Buzzer_Alarm_Driver.c

```c
/**
 ************************************************
 *@file      : Buzzer_Alarm_Driver.c
 *@author    : Mina Gamil
 *@date      : Feb 04, 2025
 *@brief     : C program for Control Buzzer States
 ************************************************
 */

/**** Include Module header ****/
#include "Buzzer_Alarm_Driver.h"

/*** Buzzer states enum ***/
enum{
    Buzzer_idle,
    Buzzer_ON,
    Buzzer_OFF
}Buzzer_State_Id;

/** Buzzer pointer to function **/
void(*Buzzer_ptr2Func)();

/** Buzzer Init API **/
void Buzzer_Init(void)
{
    Buzzer_ptr2Func = STATE(Buzzer_OFF);            // Set Buzzer Pointer To Function equal to the Address of the idle state
    Buzzer_ptr2Func();                              // Run pointed Function to Turn Off the Buzzer
}

/** Set Buzzer To ON API **/
void Set_Buzzer_To_ON()
{
    Buzzer_ptr2Func = STATE(Buzzer_ON);             // Set Buzzer Pointer To Function equal to the Address of the ON state
    Buzzer_ptr2Func();                              // Run Pointed Function
}

/** Set Buzzer To OFF API **/
void Set_Buzzer_To_OFF()
{
    Buzzer_ptr2Func = STATE(Buzzer_OFF);            // Set Buzzer Pointer To Function equal to the Address of the OFF state
    Buzzer_ptr2Func();                              // Run Pointed Function
}

/** Buzzer Idle State **/
STATE_Define(Buzzer_idle)
{
    Buzzer_State_Id = Buzzer_idle;                  // Set_State_Id equal to idle state
    Buzzer_ptr2Func = STATE(Buzzer_idle);           // Set Buzzer Pointer To Function equal to the Address of the idle state
}

/** Buzzer ON State **/
STATE_Define(Buzzer_ON)
{
    Buzzer_State_Id = Buzzer_ON;                    // Set_State_Id equal to ON state
    Set_Buzzer_Alarm(1);                            // Call Function to set buzzer to ON
    Buzzer_ptr2Func = STATE(Buzzer_idle);           // Set Buzzer Pointer To Function equal to the Address of the idle state
}

/** Buzzer OFF State **/
STATE_Define(Buzzer_OFF)
{
    Buzzer_State_Id = Buzzer_OFF;                   // Set_State_Id equal to OFF state
    Set_Buzzer_Alarm(0);                            // Call Function to set buzzer to OFF
    Buzzer_ptr2Func = STATE(Buzzer_idle);           // Set Buzzer Pointer To Function equal to the Address of the idle state
}
```

6.2.2. Buzzer_Alarm_Driver.h

```c
/*
 * Buzzer_Alarm_Driver.h
 *
 *  Created on: Feb 04, 2025
 *      Author: Mina Gamil
 */

#ifndef BUZZER_ALARM_DRIVER_H_
#define BUZZER_ALARM_DRIVER_H_

/**** Include state header ****/
#include "state.h"

/** Buzzer API's **/
void Buzzer_Init(void);
void Set_Buzzer_To_ON();
void Set_Buzzer_To_OFF();
STATE_Define(Buzzer_idle);
STATE_Define(Buzzer_ON);
STATE_Define(Buzzer_OFF);

#endif /* BUZZER_ALARM_DRIVER_H_ */
```

6.3. Light Alarm:

6.3.1. Light_Alarm_Driver.c

```c
/**
 ************************************************
 *@file      : Light_Alarm_Driver.c
 *@author    : Mina Gamil
 *@date      : Feb 04, 2025
 *@brief     : C program for Control Light States
 ************************************************
 */

/**** Include Module header ****/
#include "Light_Alarm_Driver.h"

/*** Light State Id enum ***/
enum{
    Red_On,
    Green_On,
    Yellow_On
}Light_state_Id;

/** Light pointer to function **/
void(*Light_ptr2Func)();

/** Light Init API **/
void Light_Init(void)
{
    Light_ptr2Func = STATE(Green_On);              // Set Light Pointer To Function equal to the Address of Green_On state
    Light_ptr2Func();                              // Run pointed Function to Turn on Green LED and Turn OFF others
}

/* Red LED Alarm API */
void Set_Led_To_Red()
{
    Light_ptr2Func = STATE(Red_On);                // Set Light Pointer To Function equal to the Address of Red_On state
    Light_ptr2Func();                              // Run pointed Function to Turn on Red LED and Turn OFF others
}

/* Green LED Alarm API */
void Set_Led_To_Green()
{
    Light_ptr2Func = STATE(Green_On);              // Set Light Pointer To Function equal to the Address of Green_On state
    Light_ptr2Func();                              // Run pointed Function to Turn on Green LED and Turn OFF others
}

/* Yellow LED Alarm API */
void Set_Led_To_Yellow()
{
    Light_ptr2Func = STATE(Yellow_On);             // Set Light Pointer To Function equal to the Address of Yellow_On state
    Light_ptr2Func();                              // Run pointed Function to Turn on Yellow LED and Turn OFF others
}

/* Red_On state */
STATE_Define(Red_On)
{
    Light_state_Id = Red_On;                       // Set Light_state_Id to Red_On
    Set_Light_Alarm(1, 0, 0);                      // Call set light api and set it to (r, G, Y) = (1, 0, 0)
}

/* Green_On state */
STATE_Define(Green_On)
{
    Light_state_Id = Green_On;                     // Set Light_state_Id to Green_On
    Set_Light_Alarm(0, 1, 0);                      // Call set light api and set it to (r, G, Y) = (0, 1, 0)
}

/* Yellow_On state */
STATE_Define(Yellow_On)
{
    Light_state_Id = Yellow_On;                    // Set Light_state_Id to Yellow_On
    Set_Light_Alarm(0, 0, 1);                      // Call set light api and set it to (r, G, Y) = (0, 0, 1)
}
```

6.3.2. Light_Alarm_Driver.h

```c
/*
 * Light_Alarm_Driver.h
 *
 *  Created on: Feb 04, 2025
 *      Author: Mina Gamil
 */
#ifndef LIGHT_ALARM_DRIVER_H_
#define LIGHT_ALARM_DRIVER_H_

/**** Include state header ****/
#include "state.h"

/* Light_Alarm_Driver API's */
void Light_Init(void);
void Set_Led_To_Red();
void Set_Led_To_Green();
void Set_Led_To_Yellow();
STATE_Define(Red_On);
STATE_Define(Green_On);
STATE_Define(Yellow_On);

#endif /* LIGHT_ALARM_DRIVER_H_ */
```

6.4. Indicate Crew

6.4.1. Indicate_Airplane_Crew.c

```c
/**
 ************************************************
 *@file      : Indicate_Airplane_Crew.c
 *@author    : Mina Gamil
 *@date      : Feb 03, 2025
 *@brief     : C program for Control pressure States
 ************************************************
 */

/**** Include Module header ****/
#include "Indicate_Airplane_Crew.h"

/**** Declare and assign used variable ****/
extern vuint32 pVal;
vuint32 PreviouspVal = 0;
vuint32 ThresholdpVal = 20;
vuint32 Alarm_Timer = 3600;

/*** Indicate_Airplane_Crew states enum ***/
enum{
    Indicate_crew_state,
    Waiting_state
}Indicator_state_id;

/** Indicator pointer to function **/
void (*Indicator_ptr2Func)();

/** Indicate_Airplane_Crew Init API **/
void Indicate_Airplane_Crew_Init(void)
{
    Indicator_ptr2Func = STATE(Indicate_crew_state);          // Set Indicator Pointer To Function equal to the Address of the Indicate_crew_state
}

/** Indicate_crew_state State **/
STATE_Define(Indicate_crew_state)
{
    Indicator_state_id = Indicate_crew_state;                 // Indicator_state_id equal to Indicate_crew_state
    if (pVal != PreviouspVal)                                 // First Check if the pressure value changed
    {                                                         // if yes proceed with the following
        if (pVal > ThresholdpVal)                             // if pressure value is more than Threshold
        {                                                     // if yes proceed with the following
            Set_Led_To_Red();                                 // Trun on Red LED and turn off all other LEDs
            Set_Buzzer_To_ON();                               // Turn on Buzzer
            Delay(Alarm_Timer);                               // Delay for alarm timer
        }
        else if (pVal == ThresholdpVal)                       // if pressure value is equal to Threshold
        {                                                     // if yes proceed with the following
            Set_Led_To_Yellow();                              // Turn On Yellow LED and turn off all other LEDs
            Set_Buzzer_To_OFF();                              // Turn off buzzer
        }
        else                                                  // if not both previous conditions
        {                                                     // proceed with the following
            Set_Led_To_Green();                               // Turn on Green LED and turn off all other LEDs
            Set_Buzzer_To_OFF();                              // Trun off Buzzer
        }
        PreviouspVal = pVal;                                  // Assign pVal to PreviouspVal
    }
    Indicator_ptr2Func = STATE(Waiting_state);                // Set Indicator Pointer To Function equal to the Address of the Waiting_state
}


/** Waiting State **/
STATE_Define(Waiting_state)
{
    Indicator_state_id = Waiting_state;                       // Indicator_state_id equal to Waiting_state
    Indicator_ptr2Func = STATE(Indicate_crew_state);          // Set Indicator Pointer To Function equal to the Address of the Indicate_crew_state
    Indicator_ptr2Func();                                     // Run Pointed Function
}
```

6.4.2. Indicate_Airplane_Crew.h

```c
/*
 * Indicate_Airplane_Crew.h
 *
 *  Created on: Feb 03, 2025
 *      Author: Mina Gamil
 */

#ifndef INDICATE_AIRPLANE_CREW_H_
#define INDICATE_AIRPLANE_CREW_H_

/**** Include state header ****/
#include "state.h"

/** Indicator pointer to function **/
extern void (*Indicator_ptr2Func)();

/** Indicate_Airplane_Crew API's **/
void Indicate_Airplane_Crew_Init(void);
STATE_Define(Indicate_crew_state);
STATE_Define(Waiting_state);

/* Other API's Used By this module form other files */
extern void Set_Buzzer_To_ON();
extern void Set_Buzzer_To_OFF();
extern void Set_Led_To_Red();
extern void Set_Led_To_Green();
extern void Set_Led_To_Yellow();

#endif /* INDICATE_AIRPLANE_CREW_H_ */
```

6.5. System interface Drivers:

6.5.1. Driver.c

```c
#include "driver.h"
#include <stdint.h>
#include <stdio.h>

void Delay(int nCount)
{
    for(; nCount != 0; nCount--);
}

int getPressureVal(){
    return (GPIOA_IDR & 0xFF);
}

void Set_Buzzer_Alarm(int i){
    if (i == 0){
        SET_BIT(GPIOA_ODR,12);
    }
    else if (i == 1){
        RESET_BIT(GPIOA_ODR,12);
    }
}

void Set_Light_Alarm(int R, int G, int Y){
    if (R == 1){
        RESET_BIT(GPIOA_ODR,13);
        SET_BIT(GPIOA_ODR,14);
        SET_BIT(GPIOA_ODR,15);
    }
    else if (G == 1){
        SET_BIT(GPIOA_ODR,13);
        RESET_BIT(GPIOA_ODR,14);
        SET_BIT(GPIOA_ODR,15);
    }
    else
    {
        SET_BIT(GPIOA_ODR,13);
        SET_BIT(GPIOA_ODR,14);
        RESET_BIT(GPIOA_ODR,15);
    }
}

void GPIO_INITIALIZATION(){
    SET_BIT(APB2ENR, 2);
    GPIOA_CRL &= 0xFF0FFFFF;
    GPIOA_CRL |= 0x00000000;
    GPIOA_CRH &= 0xFF0FFFFF;
    GPIOA_CRH |= 0x22222222;

}
```

6.5.2. Driver.h

```c
#include <stdint.h>
#include <stdio.h>

#ifndef BARE_METAL_H_
#define BARE_METAL_H_

#define SET_BIT(ADDRESS,BIT)   ADDRESS |=  (1<<BIT)
#define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
#define TOGGLE_BIT(ADDRESS,BIT)  ADDRESS ^=  (1<<BIT)
#define READ_BIT(ADDRESS,BIT) ((ADDRESS) &   (1<<(BIT)))

#define GPIO_PORTA 0x40010800
#define BASE_RCC   0x40021000

#define APB2ENR   *(volatile uint32_t *)(BASE_RCC + 0x18)

#define GPIOA_CRL *(volatile uint32_t *)(GPIO_PORTA + 0x00)
#define GPIOA_CRH *(volatile uint32_t *)(GPIO_PORTA + 0X04)
#define GPIOA_IDR *(volatile uint32_t *)(GPIO_PORTA + 0x08)
#define GPIOA_ODR *(volatile uint32_t *)(GPIO_PORTA + 0x0C)

void Delay(int nCount);
int getPressureVal();
void Set_Buzzer_Alarm(int i);
void Set_Light_Alarm(int R, int G, int Y);
void GPIO_INITIALIZATION();

#endif
```

6.6. startup.c:

```c
/**
 ******************************************************
 * @file        : startup.c
 * @author      : Mina Gamil
 * @date        : 3 Jan. 2025
 * @brief       : Startup file for Arm Cortex-m3
 ******************************************************
 */

/******* INCLUDE PLATFORM TYPES ********/
#include "Platform_Types.h"

/*** Declare Extern From Linker File***/
extern int main(void);
extern uint32 _stack_top;
extern uint32 _E_text;
extern uint32 _S_data;
extern uint32 _E_data;
extern uint32 _S_bss;
extern uint32 _E_bss;
uint32 i;


/*** Prototype of Reset and Default Handler ***/
void Reset_Handler();
void Default_Handler();

/** Prototype of Interrupts Handler with weak and alias **/
void NMI_Handler()__attribute__((weak, alias("Default_Handler")));
void H_Fault_Handler()__attribute__((weak, alias("Default_Handler")));
void MM_Fault_Handler()__attribute__((weak, alias("Default_Handler")));
void Bus_Fault()__attribute__((weak, alias("Default_Handler")));
void Usage_Fault_Handler()__attribute__((weak, alias("Default_Handler")));

/** vector interrupts table start with stack pointer address **/
uint32 vectors[]__attribute__((section(".vectors"))) =
{
    (uint32) &_stack_top,
    (uint32) &Reset_Handler,
    (uint32) &NMI_Handler,
    (uint32) &H_Fault_Handler,
    (uint32) &MM_Fault_Handler,
    (uint32) &Bus_Fault,
    (uint32) &Usage_Fault_Handler
};

/** Default_Handler function **/
void Default_Handler()
{
    Reset_Handler();
}

/*** Reset_Handler Function **/
void Reset_Handler()
{
    /** Get the Size of .data Section **/
    uint32 Data_Size = (uint8*)&_E_data - (uint8*)&_S_data;
    uint8* pSrc = (uint8*)&_E_text;   /* Assign Source pointer*/
    uint8* pDst = (uint8*)&_S_data;   /* Assign Destination pointer*/
    for (i = 0; i < Data_Size; i++)
    {
        /** Looping to Copy data from flash to Sram **/
        *((uint8*)pDst++) = *((uint8*)pSrc++);
    }

    /** Get the Size of .bss section **/
    uint32 Bss_Size = (uint8*)&_E_bss - (uint8*)&_S_bss;
    pDst = (uint8*)&_S_bss;           /* Re-assign Destination pointer*/
    for (i = 0; i < Bss_Size; i++)
    {
        /** Looping to keep bss storage and initialize it to Zero**/
        *((uint8*)pDst++) = (uint8) 0;
    }

    /** Jump To main() After finishing of startup steps**/
    main();
}
```

6.7. Linkerscript.ld

```
/* Learn-In-Depth
   First_Term --> First_Project --> High_Pressure_Detection_System project
   Cortex-M3 Linker
   Eng. Mina Gamil
*/

MEMORY
{
    flash(rx) : ORIGIN = 0x08000000 , LENGTH = 128k
    sram(rwx) : ORIGIN = 0x20000000 , LENGTH = 20k
}

SECTIONS
{
    .text : {
        *(.vectors*)
        *(.text*)
        *(.rodata)
        _E_text = .;             /* Symbol of End of Text in flash - Symbol of Start of data in flash */
    }> flash

    .data : {
        _S_data = .;            /* Symbol of Start of .data section in Sram*/
        *(.data)
        _E_data = .;            /* Symbol of End of .data Section in Sram */
    }> sram AT>flash

    .bss : {
        _S_bss = .,            /* Symbol of Start of .bss section in Sram */
        *(.bss)
        . = ALIGN(4);
        _E_bss = .,            /* Symbol of End of .bss section in Sram */
        . = . + 0x1000;        /* Create Stack Area */
        _stack_top = .;        /* Assign last Address to stack pointer Address symbol */
    }> sram
}
```

6.8. state.h

```
/*
 * state.h
 *
 *  Created on: Feb 03, 2025
 *      Author: Mina Gamil
 */

#ifndef STATE_H_
#define STATE_H_

/** Include Needed STD Library **/
#include <stdio.h>
#include <stdlib.h>
#include "Platform_Types.h"
#include "driver.h"

/** Define Needed Macros to create states automatically **/
#define STATE_Define(_stateFunc_) void ST_##_stateFunc_()
#define STATE(_stateFunc_) ST_##_stateFunc_


#endif /* STATE_H_ */
```

6.9. Platform_Types.h

```c
/*
 ************************************************************
 * @file         : Platform_Types.h
 * @author       : Mina Gamil
 * @date         : 24 Dec. 2024
 * @brief        : Platform Types Definitions Header File
 ************************************************************
*/

#ifndef PLATFORM_TYPES_H_
#define PLATFORM_TYPES_H_

/************** TRUE-FALSE DEFINITION *************/
#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

/************* BOOLEAN DATA TYPE *************/
typedef unsigned char       boolean;

/******************** UNSIGNED DATA TYPES ********************/
typedef unsigned char                   uint8;          // 0..255
typedef unsigned short                  uint16;         // 0..65535
typedef unsigned int                    uint32;         // 0..4294967295
typedef signed long long                uint64;         // 0..18446744073709551615

typedef unsigned char                   uint8_least;    // At Least 0..255
typedef unsigned short                  uint16_least;   // At Least 0..65535
typedef unsigned int                    uint32_least;   // At Least 0..4294967295

typedef volatile unsigned char          vuint8;         // 0..255
typedef volatile unsigned short         vuint16;        // 0..65535
typedef volatile unsigned int           vuint32;        // 0..4294967295
typedef volatile unsigned long long     vuint64;        // 0..18446744073709551615

/******************** SIGNED DATA TYPES ********************/
typedef signed char                     sint8;          // -127..+127
typedef signed short                    sint16;         // -32768..+32767
typedef signed int                      sint32;         // -2147483648..+2147483647
typedef signed long long                sint64;         // -9223372036854775808..+9223372036854775807

typedef signed char                     sint8_least;    // At Least -127..+127
typedef signed short                    sint16_least;   // At Least -32768..+32767
typedef signed int                      sint32_least;   // At Least -2147483648..+2147483647

typedef volatile signed char            vsint8;         // -127..+127
typedef volatile signed short           vsint16;        // -32768..+32767
typedef volatile signed int             vsint32;        // -2147483648..+2147483647
typedef volatile signed long long       vsint64;        // -9223372036854775808..+9223372036854775807

typedef float                           float32;        // -3.4028235e+38..+3.4028235e+38
typedef double                          float64;        // -1.7976931348623157e+308..+1.7976931348623157e+308

typedef volatile float                  vfloat32;       // -3.4028235e+38..+3.4028235e+38
typedef volatile double                 vfloat64;       // -1.7976931348623157e+308..+1.7976931348623157e+308

/******************** VOID POINTERS ********************/
typedef void*                           VoidPtr;        // Void Pointer
typedef const void*                     ConstVoidPtr;   // Contant Void Pointer


#endif /* PLATFORM_TYPES_H_ */
```

6.10. main.c

```c
/**
 ********************************************
 *@file      : main.c
 *@author    : Mina Gamil
 *@date      : Feb 03, 2025
 *@brief     : C program for Pressure Detection
 ********************************************
 */

/* Included Needed Libraries */
#include "driver.h"
#include "Pressure_Sensor_Driver.h"
#include "Indicate_Airplane_Crew.h"

/* Void setup for init or one time run code */
void setup(void)
{
    GPIO_INITIALIZATION();                  // GPIO Init
    Pressure_Sensor_Init();                 // Sensor Init
    Buzzer_Init();                          // Buzzer Init
    Light_Init();                           // Light Init
    Indicate_Airplane_Crew_Init();          // Indicator Init
}


/* Main Program */
int main(void)
{
    setup();
    while(1)                                // Infinite loop
    {
        Pressure_Sensor_ptr2Func();         // Run Function pointed by Pressure_Sensor_ptr2Func
        Delay(10);                          // Wait for 10 millisecond
        Indicator_ptr2Func();               // Run Function pointed by Indicator_ptr2Func
        Delay(10);                          // Wait for 10 millisecond
    }
    return 0;
}
```

6.11. makefile

```
#Makefile for High_Pressure_Detection_System

# Declare Macros to make Generic Makefile
CC           = arm-none-eabi-
CFLAGS       = -mcpu=cortex-m3 -mthumb -gdwarf-2
INCS         = -I .
LIBS         =
SRC          = $(wildcard *.c)
OBJ          = $(SRC:.c=.o)
AS           = $(wildcard *.s)
ASOBJ        = $(AS:.s=.o)
Project_Name = main

all:$(Project_Name).hex
    @echo "********** Build is Done **********"

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

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

$(Project_Name).elf:$(ASOBJ) $(OBJ)
    $(CC)ld -T linker_script.ld $(LIBS) $(ASOBJ) $(OBJ) -o $@ -M=Map_file.map

$(Project_Name).hex:$(Project_Name).elf
    $(CC)objcopy -O binary $< $@


clean:
    rm *.bin *elf *.hex
    @echo "*** All .bin/.elf files deleted ***"

clean_all:
    rm *.o *.bin *.elf *.map *.hex
    @echo "*** All built files deleted ***"
```

7. Software analysis:

7.1. Section Table

7.1.1. Pressure_Sensor_Driver.o:

```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_System
line_Diploma/First_Term/First_Project/High_Pressure_Detection_System (main)
$ arm-none-eabi-objdump -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          00000000  00000000  00000000  000000bc  2**0
                  ALLOC
  3 .debug_info   0000012f  00000000  00000000  000000bc  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000a6  00000000  00000000  000001eb  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000084  00000000  00000000  00000291  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000315  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000068  00000000  00000000  00000335  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000001f1  00000000  00000000  0000039d  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  0000058e  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000005a0  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000060  00000000  00000000  000005d4  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

7.1.2. Buzzer_Alarm_Driver.o:

```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_System
line_Diploma/First_Term/First_Project/High_Pressure_Detection_System (main)
$ arm-none-eabi-objdump -h Buzzer_Alarm_Driver.o

Buzzer_Alarm_Driver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000f8  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000012c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000012c  2**0
                  ALLOC
  3 .debug_info   00000153  00000000  00000000  0000012c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000aa  00000000  00000000  0000027f  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000108  00000000  00000000  00000329  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000431  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000005e  00000000  00000000  00000451  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000001dd  00000000  00000000  000004af  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  0000068c  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  0000069e  2**0
                  CONTENTS, READONLY
 11 .debug_frame  000000b4  00000000  00000000  000006d4  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

7.1.3. Light_Alarm_Driver.o:

```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_System
oma/First_Term/First_Project/High_Pressure_Detection_System (main)
$ arm-none-eabi-objdump -h Light_Alarm_Driver.o

Light_Alarm_Driver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000fc  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000130  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000130  2**0
                  ALLOC
  3 .debug_info   00000168  00000000  00000000  00000130  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 00000094  00000000  00000000  00000298  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000134  00000000  00000000  0000032c  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000460  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000005f  00000000  00000000  00000480  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000001da  00000000  00000000  000004df  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  000006b9  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000006cb  2**0
                  CONTENTS, READONLY
 11 .debug_frame  000000d4  00000000  00000000  00000700  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

7.1.4. Indicate_Airplane_Crew.o:

```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_System
Mina@Bello MINGW64 /d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Dipl
oma/First_Term/First_Project/High_Pressure_Detection_System (main)
$ arm-none-eabi-objdump -h Indicate_Airplane_Crew.o

Indicate_Airplane_Crew.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000100  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000008  00000000  00000000  00000134  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  0000013c  2**2
                  ALLOC
  3 .debug_info   00000161  00000000  00000000  0000013c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000b9  00000000  00000000  0000029d  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000084  00000000  00000000  00000356  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  000003da  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000072  00000000  00000000  000003fa  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000206  00000000  00000000  0000046c  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  00000672  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00000684  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000060  00000000  00000000  000006b8  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

7.1.5. driver.o:

```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_System
oma/First_Term/First_Project/High_Pressure_Detection_System (main)
$ arm-none-eabi-objdump -h driver.o

driver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000020c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000240  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000240  2**0
                  ALLOC
  3 .debug_info   00000142  00000000  00000000  00000240  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 0000009d  00000000  00000000  00000382  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000100  00000000  00000000  0000041f  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  0000051f  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000000a6  00000000  00000000  0000053f  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000183  00000000  00000000  000005e5  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  00000768  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  0000077a  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000094  00000000  00000000  000007b0  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

7.1.6. startup.o:

```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_System
Mina@Bello MINGW64 /d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Dipl
oma/First_Term/First_Project/High_Pressure_Detection_System (main)
$ arm-none-eabi-objdump -h startup.o

startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000fc  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000130  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000130  2**0
                  ALLOC
  3 .vectors      0000001c  00000000  00000000  00000130  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  4 .debug_info   00000176  00000000  00000000  0000014c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 000000c2  00000000  00000000  000002c2  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    00000064  00000000  00000000  00000384  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  000003e8  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_line   0000007b  00000000  00000000  00000408  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  9 .debug_str    00000198  00000000  00000000  00000483  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .comment      00000012  00000000  00000000  0000061b  2**0
                  CONTENTS, READONLY
 11 .ARM.attributes 00000033  00000000  00000000  0000062d  2**0
                  CONTENTS, READONLY
 12 .debug_frame  0000004c  00000000  00000000  00000660  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

7.1.7. main.o:

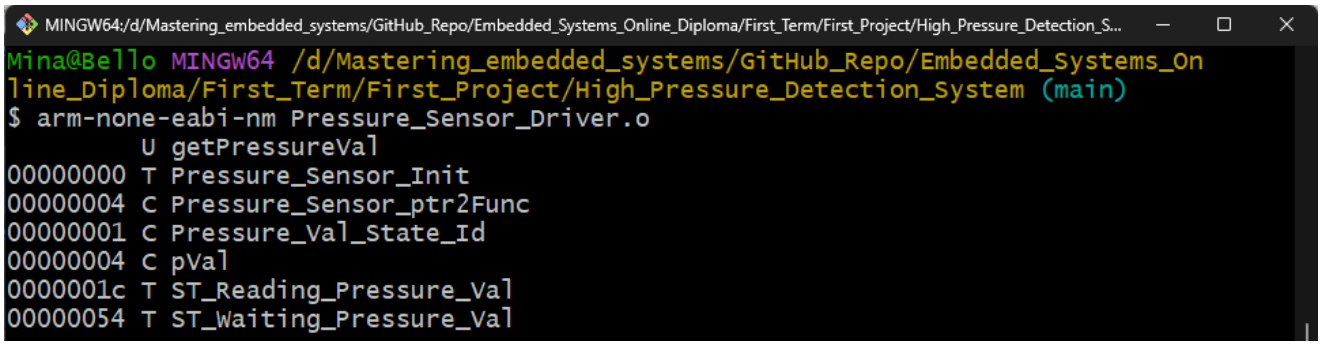
```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_S...
Mina@Bello MINGW64 /d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_On
line_Diploma/First_Term/First_Project/High_Pressure_Detection_System (main)
$ arm-none-eabi-objdump -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000050  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000084  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000084  2**0
                  ALLOC
  3 .debug_info   00000106  00000000  00000000  00000084  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000000a8  00000000  00000000  0000018a  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000058  00000000  00000000  00000232  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  0000028a  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000008f  00000000  00000000  000002aa  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000182  00000000  00000000  00000339  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  000004bb  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000004cd  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000048  00000000  00000000  00000500  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

7.1.8. main.elf

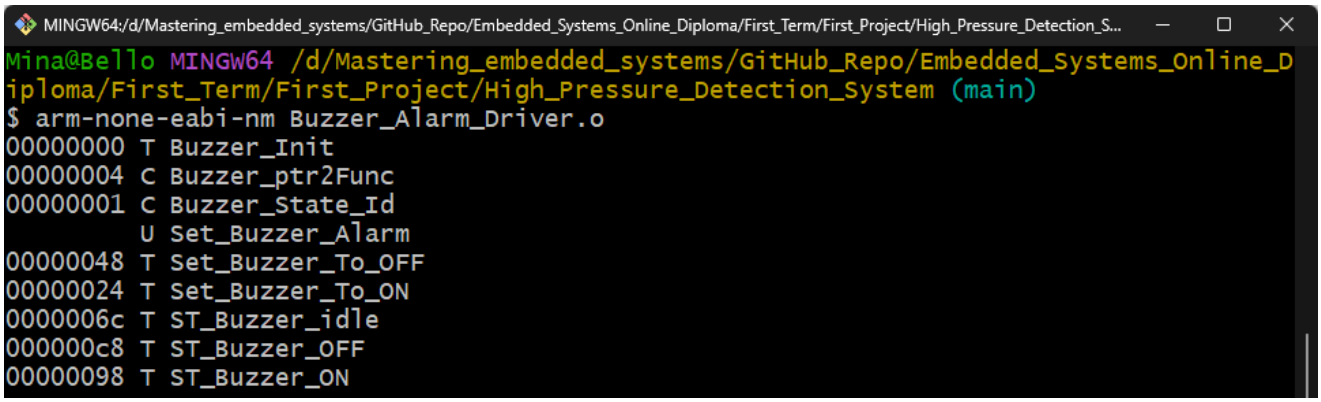
```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_S...
Mina@Bello MINGW64 /d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_D
iploma/First_Term/First_Project/High_Pressure_Detection_System (main)
$ arm-none-eabi-objdump -h main.elf

main.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000006f0  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  20000000  080006f0  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          0000102c  20000008  080006f8  00010008  2**2
                  ALLOC
  3 .debug_info   00000909  00000000  00000000  00010008  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 000004a4  00000000  00000000  00010911  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000500  00000000  00000000  00010db5  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000e0  00000000  00000000  000112b5  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000347  00000000  00000000  00011395  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    0000044e  00000000  00000000  000116dc  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  00011b2a  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00011b3b  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000370  00000000  00000000  00011b70  2**2
                  CONTENTS, READONLY, DEBUGGING
```

7.2. Symbols table:

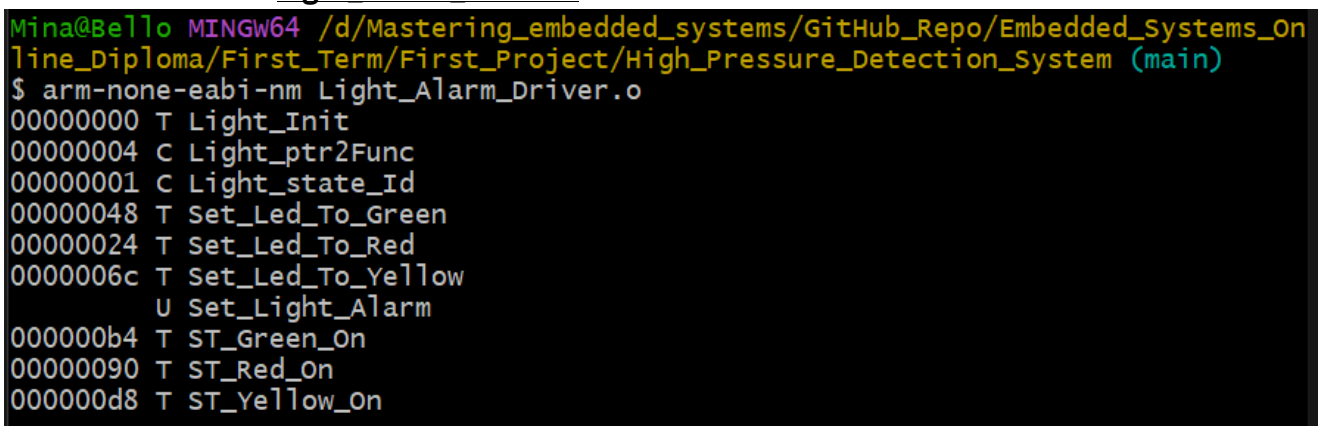
7.2.1. Pressure_Sensor_Driver.o



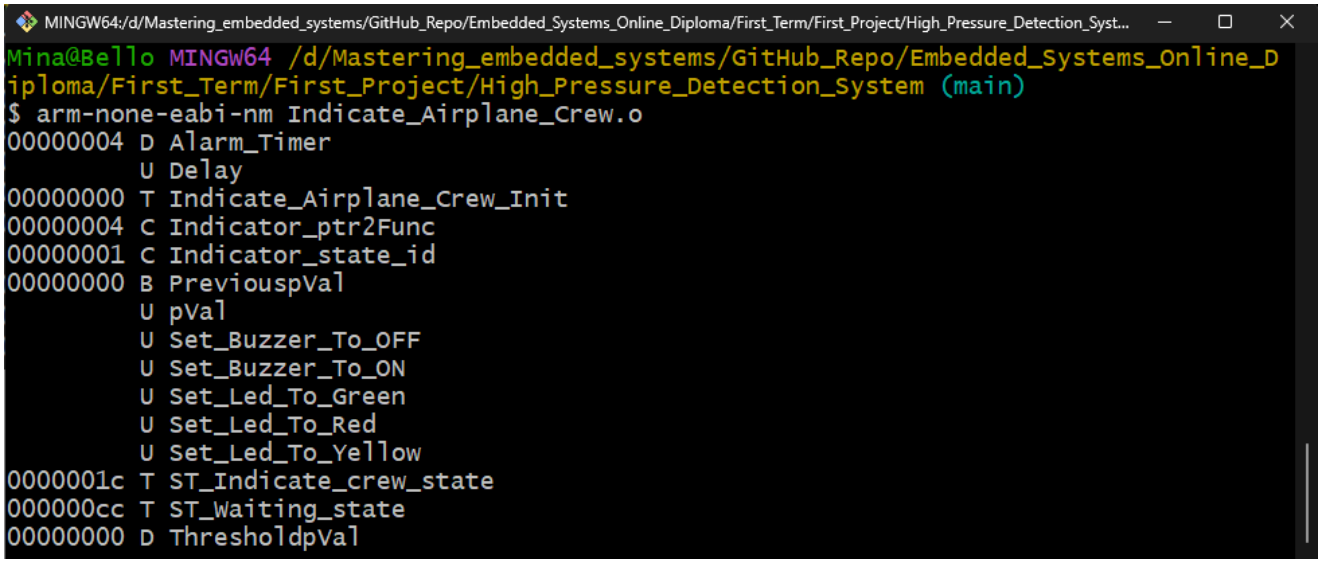
7.2.2. Buzzer_Alarm_Driver.o



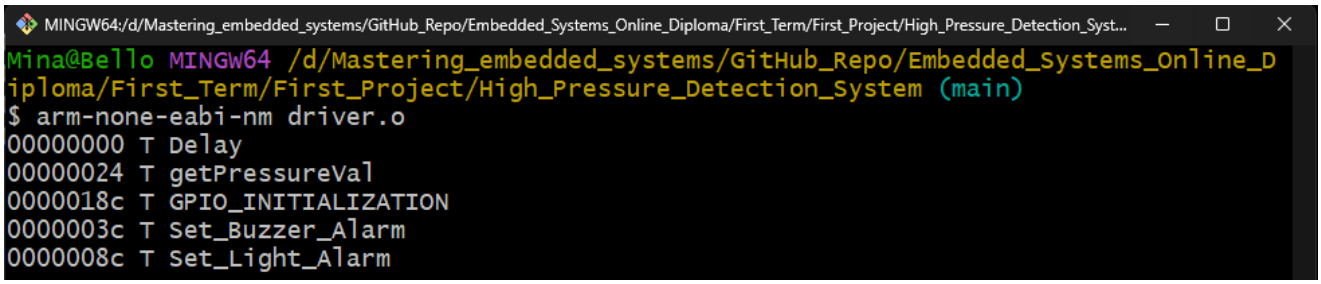
7.2.3. Light_Alarm_Driver.o



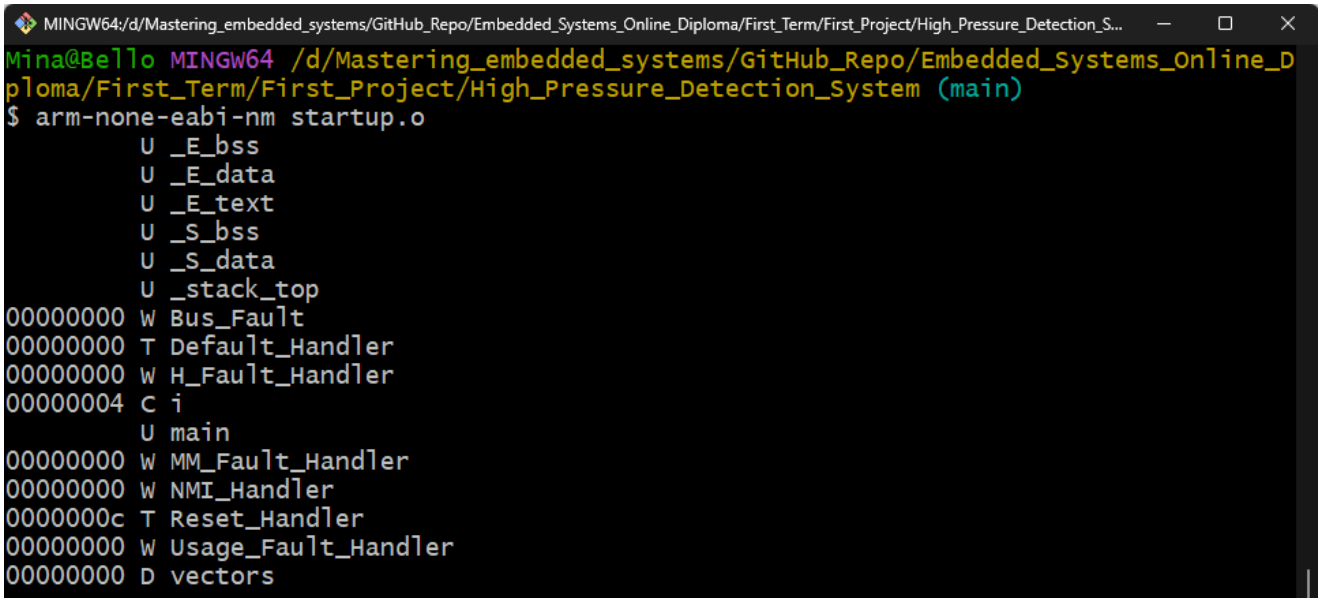
7.2.4. Indicate_Airplane_Crew.o



7.2.5. driver.o



7.2.6. startup.o



7.2.7. main.o

```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_S...
Mina@Bello MINGW64 /d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_D
ploma/First_Term/First_Project/High_Pressure_Detection_System (main)
$ arm-none-eabi-nm main.o
         U Buzzer_Init
         U Delay
         U GPIO_INITIALIZATION
         U Indicate_Airplane_Crew_Init
         U Indicator_ptr2Func
         U Light_Init
0000001c T main
         U Pressure_Sensor_Init
         U Pressure_Sensor_ptr2Func
00000000 T setup
```

7.2.8. main.elf:

```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_S...
Mina@Bello MINGW64 /d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_D
ploma/First_Term/First_Project/High_Pressure_Detection_System (main)
$ arm-none-eabi-nm main.elf
2000000c B _E_bss
20000008 D _E_data
080006f0 T _E_text
20000008 B _S_bss
20000000 D _S_data
2000100c B _stack_top
20000004 D Alarm_Timer
080005f4 W Bus_Fault
0800001c T Buzzer_Init
20001010 B Buzzer_ptr2Func
2000100c B Buzzer_State_Id
080005f4 T Default_Handler
08000114 T Delay
08000138 T getPressureVal
080002a0 T GPIO_INITIALIZATION
080005f4 W H_Fault_Handler
20001030 B i
08000320 T Indicate_Airplane_Crew_Init
20001018 B Indicator_ptr2Func
20001014 B Indicator_state_id
08000420 T Light_Init
2000101c B Light_ptr2Func
20001020 B Light_state_Id
08000538 T main
080005f4 W MM_Fault_Handler
080005f4 W NMI_Handler
0800056c T Pressure_Sensor_Init
20001024 B Pressure_Sensor_ptr2Func
2000102c B Pressure_Val_State_Id
20000008 B PreviouspVal
20001028 B pVal
08000600 T Reset_Handler
08000150 T Set_Buzzer_Alarm
08000064 T Set_Buzzer_To_OFF
08000040 T Set_Buzzer_To_ON
08000468 T Set_Led_To_Green
08000444 T Set_Led_To_Red
0800048c T Set_Led_To_Yellow
080001a0 T Set_Light_Alarm
0800051c T setup
08000088 T ST_Buzzer_idle
080000e4 T ST_Buzzer_OFF
080000b4 T ST_Buzzer_ON
080004d4 T ST_Green_On
0800033c T ST_Indicate_crew_state
08000588 T ST_Reading_Pressure_Val
080004b0 T ST_Red_On
080005c0 T ST_Waiting_Pressure_Val
080003ec T ST_Waiting_state
080004f8 T ST_Yellow_On
20000000 D ThresholdpVal
080005f4 W Usage_Fault_Handler
08000000 T vectors
```

7.3. Mapfile.map:

D:\Mastering_embedded_systems\GitHub_Repo\Embedded_Systems_Online_Diploma\First_Term\First_Project\High...

File Edit Selection Find View Goto Tools Project Preferences Help

Map_file.map

```
Allocating common symbols
Common symbol       size              file

Indicator_state_id  0x1               Indicate_Airplane_Crew.o
Pressure_Sensor_ptr2Func
                    0x4               Pressure_Sensor_Driver.o
pVal                0x4               Pressure_Sensor_Driver.o
Indicator_ptr2Func  0x4               Indicate_Airplane_Crew.o
Light_ptr2Func      0x4               Light_Alarm_Driver.o
Buzzer_State_Id     0x1               Buzzer_Alarm_Driver.o
i                   0x4               startup.o
Buzzer_ptr2Func     0x4               Buzzer_Alarm_Driver.o
Light_state_Id      0x1               Light_Alarm_Driver.o
Pressure_Val_State_Id
                    0x1               Pressure_Sensor_Driver.o

Memory Configuration

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

Linker script and memory map


.text           0x08000000      0x6f0
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x0800001c       0xf8 Buzzer_Alarm_Driver.o
                0x0800001c                Buzzer_Init
                0x08000040                Set_Buzzer_To_ON
                0x08000064                Set_Buzzer_To_OFF
                0x08000088                ST_Buzzer_idle
                0x080000b4                ST_Buzzer_ON
                0x080000e4                ST_Buzzer_OFF
 .text          0x08000114      0x20c driver.o
                0x08000114                Delay
                0x08000138                getPressureVal
                0x08000150                Set_Buzzer_Alarm
                0x080001a0                Set_Light_Alarm
                0x080002a0                GPIO_INITIALIZATION
 .text          0x08000320      0x100 Indicate_Airplane_Crew.o
                0x08000320                Indicate_Airplane_Crew_Init
                0x0800033c                ST_Indicate_crew_state
```

```
                0x080003ec                ST_Waiting_state
 .text          0x08000420       0xfc Light_Alarm_Driver.o
                0x08000420                Light_Init
                0x08000444                Set_Led_To_Red
                0x08000468                Set_Led_To_Green
                0x0800048c                Set_Led_To_Yellow
                0x080004b0                ST_Red_On
                0x080004d4                ST_Green_On
                0x080004f8                ST_Yellow_On
 .text          0x0800051c       0x50 main.o
                0x0800051c                setup
                0x08000538                main
 .text          0x0800056c       0x88 Pressure_Sensor_Driver.o
                0x0800056c                Pressure_Sensor_Init
                0x08000588                ST_Reading_Pressure_Val
                0x080005c0                ST_Waiting_Pressure_Val
 .text          0x080005f4       0xfc startup.o
                0x080005f4                H_Fault_Handler
                0x080005f4                MM_Fault_Handler
                0x080005f4                Usage_Fault_Handler
                0x080005f4                Bus_Fault
                0x080005f4                Default_Handler
                0x080005f4                NMI_Handler
                0x08000600                Reset_Handler
 *(.rodata)
                0x080006f0                _E_text = .

 .glue_7        0x080006f0        0x0
  .glue_7       0x00000000        0x0 linker stubs

 .glue_7t       0x080006f0        0x0
  .glue_7t      0x00000000        0x0 linker stubs

 .vfp11_veneer  0x080006f0        0x0
  .vfp11_veneer 0x00000000        0x0 linker stubs

 .v4_bx         0x080006f0        0x0
  .v4_bx        0x00000000        0x0 linker stubs

 .iplt          0x080006f0        0x0
  .iplt         0x00000000        0x0 Buzzer_Alarm_Driver.o

 .rel.dyn       0x080006f0        0x0
  .rel.iplt     0x00000000        0x0 Buzzer_Alarm_Driver.o

 .data          0x20000000        0x8 load address 0x080006f0
                0x20000000                _S_data = .
 *(.data)
 .data          0x20000000        0x0 Buzzer_Alarm_Driver.o
 .data          0x20000000        0x0 driver.o
 .data          0x20000000        0x8 Indicate_Airplane_Crew.o
                0x20000000                ThresholdpVal
                0x20000004                Alarm_Timer
 .data          0x20000008        0x0 Light_Alarm_Driver.o
```

```
 .data           0x20000008        0x0 main.o
 .data           0x20000008        0x0 Pressure_Sensor_Driver.o
 .data           0x20000008        0x0 startup.o
                 0x20000008                _E_data = .

.igot.plt        0x20000008        0x0 load address 0x080006f8
 .igot.plt       0x00000000        0x0 Buzzer_Alarm_Driver.o

.bss             0x20000008     0x102c load address 0x080006f8
                 0x20000008                _S_bss = .
 *(.bss)
 .bss            0x20000008        0x0 Buzzer_Alarm_Driver.o
 .bss            0x20000008        0x0 driver.o
 .bss            0x20000008        0x4 Indicate_Airplane_Crew.o
                 0x20000008                PreviouspVal
 .bss            0x2000000c        0x0 Light_Alarm_Driver.o
 .bss            0x2000000c        0x0 main.o
 .bss            0x2000000c        0x0 Pressure_Sensor_Driver.o
 .bss            0x2000000c        0x0 startup.o
                 0x2000000c                . = ALIGN (0x4)
                 0x2000000c                _E_bss = .
                 0x2000100c                . = (. + 0x1000)
 *fill*          0x2000000c     0x1000
                 0x2000100c                _stack_top = .
 COMMON          0x2000100c        0x8 Buzzer_Alarm_Driver.o
                 0x2000100c                Buzzer_State_Id
                 0x20001010                Buzzer_ptr2Func
 COMMON          0x20001014        0x8 Indicate_Airplane_Crew.o
                 0x20001014                Indicator_state_id
                 0x20001018                Indicator_ptr2Func
 COMMON          0x2000101c        0x5 Light_Alarm_Driver.o
                 0x2000101c                Light_ptr2Func
                 0x20001020                Light_state_Id
 *fill*          0x20001021        0x3
 COMMON          0x20001024        0x9 Pressure_Sensor_Driver.o
                 0x20001024                Pressure_Sensor_ptr2Func
                 0x20001028                pVal
                 0x2000102c                Pressure_Val_State_Id
 *fill*          0x2000102d        0x3
 COMMON          0x20001030        0x4 startup.o
                 0x20001030                i
LOAD Buzzer_Alarm_Driver.o
LOAD driver.o
LOAD Indicate_Airplane_Crew.o
LOAD Light_Alarm_Driver.o
LOAD main.o
LOAD Pressure_Sensor_Driver.o
LOAD startup.o
OUTPUT(main.elf elf32-littlearm)

.debug_info      0x00000000      0x909
 .debug_info     0x00000000      0x153 Buzzer_Alarm_Driver.o
 .debug_info     0x00000153      0x142 driver.o
 .debug_info     0x00000295      0x161 Indicate_Airplane_Crew.o
 .debug_info     0x000003f6      0x168 Light_Alarm_Driver.o
```

```
 .debug_info    0x0000055e      0x106 main.o
 .debug_info    0x00000664      0x12f Pressure_Sensor_Driver.o
 .debug_info    0x00000793      0x176 startup.o

.debug_abbrev   0x00000000      0x4a4
 .debug_abbrev  0x00000000       0xaa Buzzer_Alarm_Driver.o
 .debug_abbrev  0x000000aa       0x9d driver.o
 .debug_abbrev  0x00000147       0xb9 Indicate_Airplane_Crew.o
 .debug_abbrev  0x00000200       0x94 Light_Alarm_Driver.o
 .debug_abbrev  0x00000294       0xa8 main.o
 .debug_abbrev  0x0000033c       0xa6 Pressure_Sensor_Driver.o
 .debug_abbrev  0x000003e2       0xc2 startup.o

.debug_loc      0x00000000      0x500
 .debug_loc     0x00000000      0x108 Buzzer_Alarm_Driver.o
 .debug_loc     0x00000108      0x100 driver.o
 .debug_loc     0x00000208       0x84 Indicate_Airplane_Crew.o
 .debug_loc     0x0000028c      0x134 Light_Alarm_Driver.o
 .debug_loc     0x000003c0       0x58 main.o
 .debug_loc     0x00000418       0x84 Pressure_Sensor_Driver.o
 .debug_loc     0x0000049c       0x64 startup.o

.debug_aranges  0x00000000       0xe0
 .debug_aranges
                0x00000000       0x20 Buzzer_Alarm_Driver.o
 .debug_aranges
                0x00000020       0x20 driver.o
 .debug_aranges
                0x00000040       0x20 Indicate_Airplane_Crew.o
 .debug_aranges
                0x00000060       0x20 Light_Alarm_Driver.o
 .debug_aranges
                0x00000080       0x20 main.o
 .debug_aranges
                0x000000a0       0x20 Pressure_Sensor_Driver.o
 .debug_aranges
                0x000000c0       0x20 startup.o

.debug_line     0x00000000      0x347
 .debug_line    0x00000000       0x5e Buzzer_Alarm_Driver.o
 .debug_line    0x0000005e       0xa6 driver.o
 .debug_line    0x00000104       0x72 Indicate_Airplane_Crew.o
 .debug_line    0x00000176       0x5f Light_Alarm_Driver.o
 .debug_line    0x000001d5       0x8f main.o
 .debug_line    0x00000264       0x68 Pressure_Sensor_Driver.o
 .debug_line    0x000002cc       0x7b startup.o

.debug_str      0x00000000      0x44e
 .debug_str     0x00000000      0x183 Buzzer_Alarm_Driver.o
                                0x1dd (size before relaxing)
 .debug_str     0x00000183       0x63 driver.o
                                0x183 (size before relaxing)
 .debug_str     0x000001e6       0xb2 Indicate_Airplane_Crew.o
                                0x206 (size before relaxing)
 .debug_str     0x00000298       0x93 Light_Alarm_Driver.o
```

```
                                0x1da (size before relaxing)
 .debug_str     0x0000032b      0x2b main.o
                                0x182 (size before relaxing)
 .debug_str     0x00000356      0x74 Pressure_Sensor_Driver.o
                                0x1f1 (size before relaxing)
 .debug_str     0x000003ca      0x84 startup.o
                                0x198 (size before relaxing)

.comment        0x00000000      0x11
 .comment       0x00000000      0x11 Buzzer_Alarm_Driver.o
                                0x12 (size before relaxing)
 .comment       0x00000000      0x12 driver.o
 .comment       0x00000000      0x12 Indicate_Airplane_Crew.o
 .comment       0x00000000      0x12 Light_Alarm_Driver.o
 .comment       0x00000000      0x12 main.o
 .comment       0x00000000      0x12 Pressure_Sensor_Driver.o
 .comment       0x00000000      0x12 startup.o

.ARM.attributes
                0x00000000      0x33
 .ARM.attributes
                0x00000000      0x33 Buzzer_Alarm_Driver.o
 .ARM.attributes
                0x00000033      0x33 driver.o
 .ARM.attributes
                0x00000066      0x33 Indicate_Airplane_Crew.o
 .ARM.attributes
                0x00000099      0x33 Light_Alarm_Driver.o
 .ARM.attributes
                0x000000cc      0x33 main.o
 .ARM.attributes
                0x000000ff      0x33 Pressure_Sensor_Driver.o
 .ARM.attributes
                0x00000132      0x33 startup.o

.debug_frame    0x00000000      0x370
 .debug_frame   0x00000000      0xb4 Buzzer_Alarm_Driver.o
 .debug_frame   0x000000b4      0x94 driver.o
 .debug_frame   0x00000148      0x60 Indicate_Airplane_Crew.o
 .debug_frame   0x000001a8      0xd4 Light_Alarm_Driver.o
 .debug_frame   0x0000027c      0x48 main.o
 .debug_frame   0x000002c4      0x60 Pressure_Sensor_Driver.o
 .debug_frame   0x00000324      0x4c startup.o
```

8. System Simulation:

8.1. Run Code on Windows:-

8.1.1. Initialization:

```
MINGW64:/d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_...
Mina@Bello MINGW64 /d/Mastering_embedded_systems/GitHub_Repo/Embedded_Systems_Online_Diploma/First_Term/First_Project/High_Pressure_Detection_System_C_Implementation (main)
$ ./main
Pressure_Sensor Initialization
Buzzer Initialization
Buzzer_Alarm is OFF
Light_Alarm Initialization
Green_LED_Alarm is ON
Indicate_Airplane_Crew Initialization
-----------------------------------
```

8.1.2. if Pressure > 20 bars

```
-----------------------------------
Pressure Got from sensor = 23 bar
Pressure Detected Higher than Threshold pressure
Red_LED_Alarm is ON
Buzzer_Alarm is ON
-----------------------------------
```

8.1.3. If Pressure = 20 bars

```
-----------------------------------
Pressure Got from sensor = 20 bar
Pressure Detected is Equal to Threshold pressure
Yellow_LED_Alarm is ON
Buzzer_Alarm is OFF
-----------------------------------
```

8.1.4. If Pressure < 20 bars

```
-----------------------------------
Pressure Got from sensor = 19 bar
Pressure Detected lower than Threshold pressure
Green_LED_Alarm is ON
Buzzer_Alarm is OFF
-----------------------------------
```

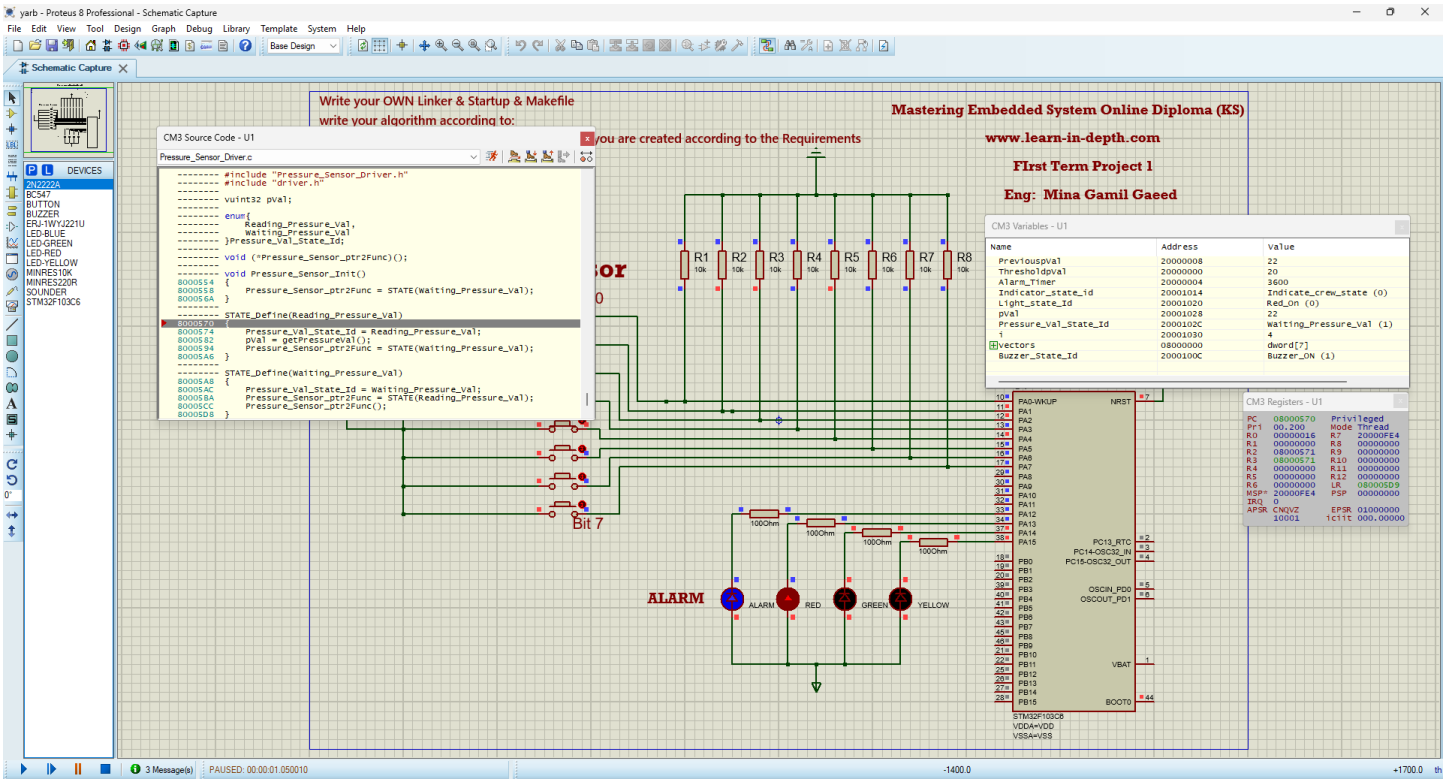
8.1.5. If pressure not change from check to another

```
-----------------------------------
Pressure Got from sensor = 19 bar
Pressure Detected lower than Threshold pressure
Green_LED_Alarm is ON
Buzzer_Alarm is OFF
-----------------------------------
Pressure Got from sensor = 19 bar
        ---------------
        Pressure Stable
        ---------------
-----------------------------------
```

8.2. Proteus simulation:

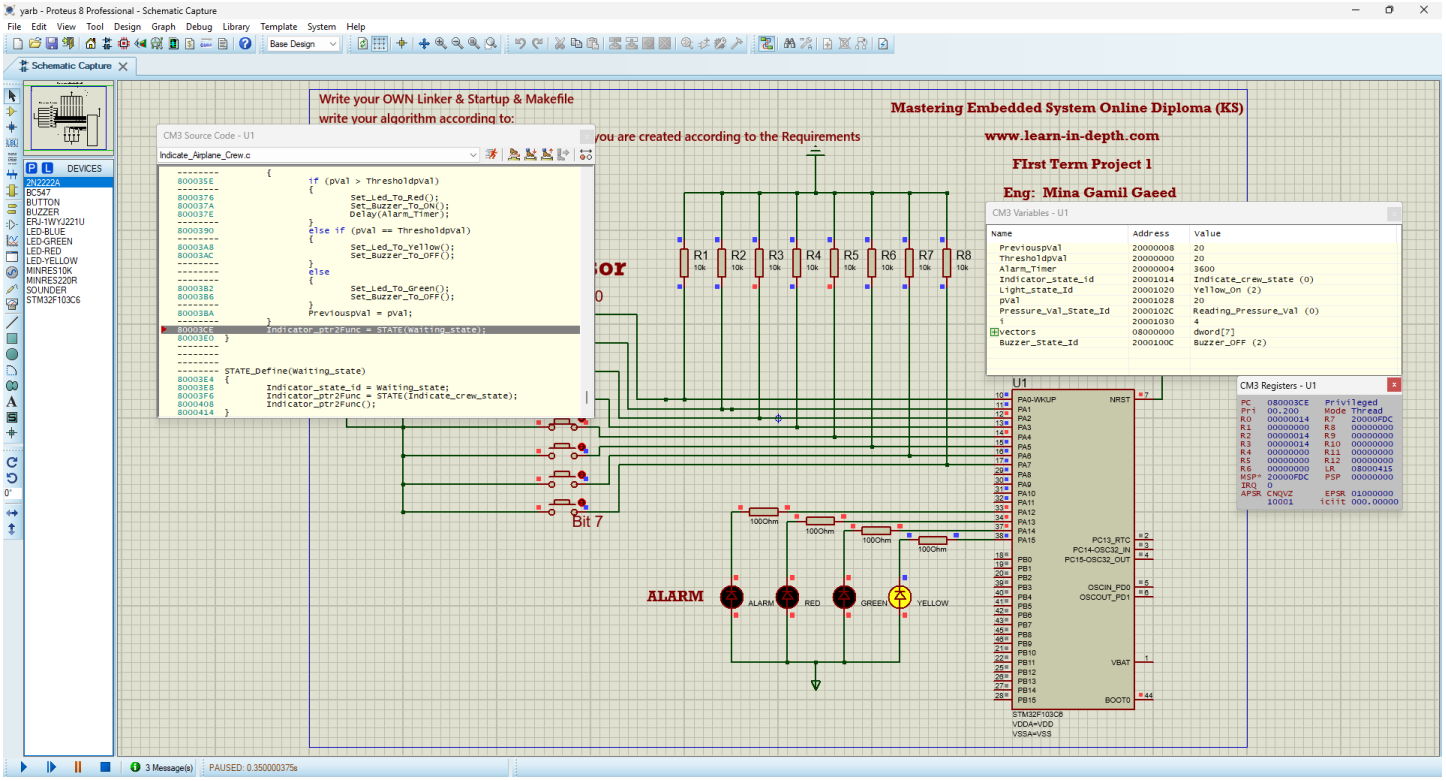
8.2.1. If pressure > 20 bars:

Red LED Will Turned ON, and Buzzer will Turned ON for 60 seconds Then check again if the state changes, stop the alarm and turn of Red LED and Turn on the LED of the Other state, if not change keep the everything as is till pressure changes.



8.2.2. If pressure = 20 bars:

Yellow LED will Turn ON.



8.2.3. If pressure < 20 bars:

Green LED will Turn ON.