Pressure Detection

Report

Mastering Embedded System Online Diploma

Prepared By : Sherif Ashraf Khedr

My Profile Link

Table Of Content

Table Of Figure

1 - Client Requirement

A ”client” expects to deliver the software of the following system specification (from the client) A pressure controller informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin the alarm duration equals 60 seconds.

1.1 - Requirement Diagram

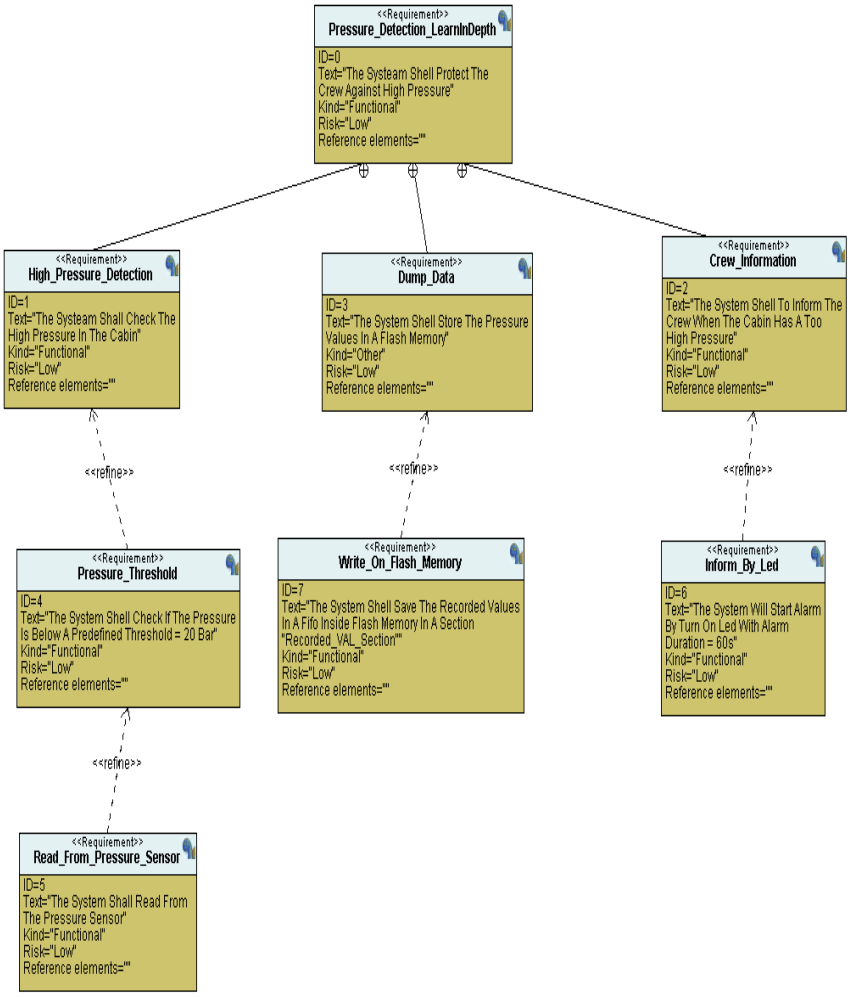


Figure 1 : Requirement Diagram

2 - Methodology

Agile is a flexible and collaborative project management approach that prioritizes iterative development, adaptability to change, and continuous improvement. It fosters regular communication among team members and stakeholders, delivering small, functional increments to meet evolving requirements efficiently. Agile is not just a methodology but a mindset that values individuals, interactions, and the delivery of working solutions in dynamic environments.

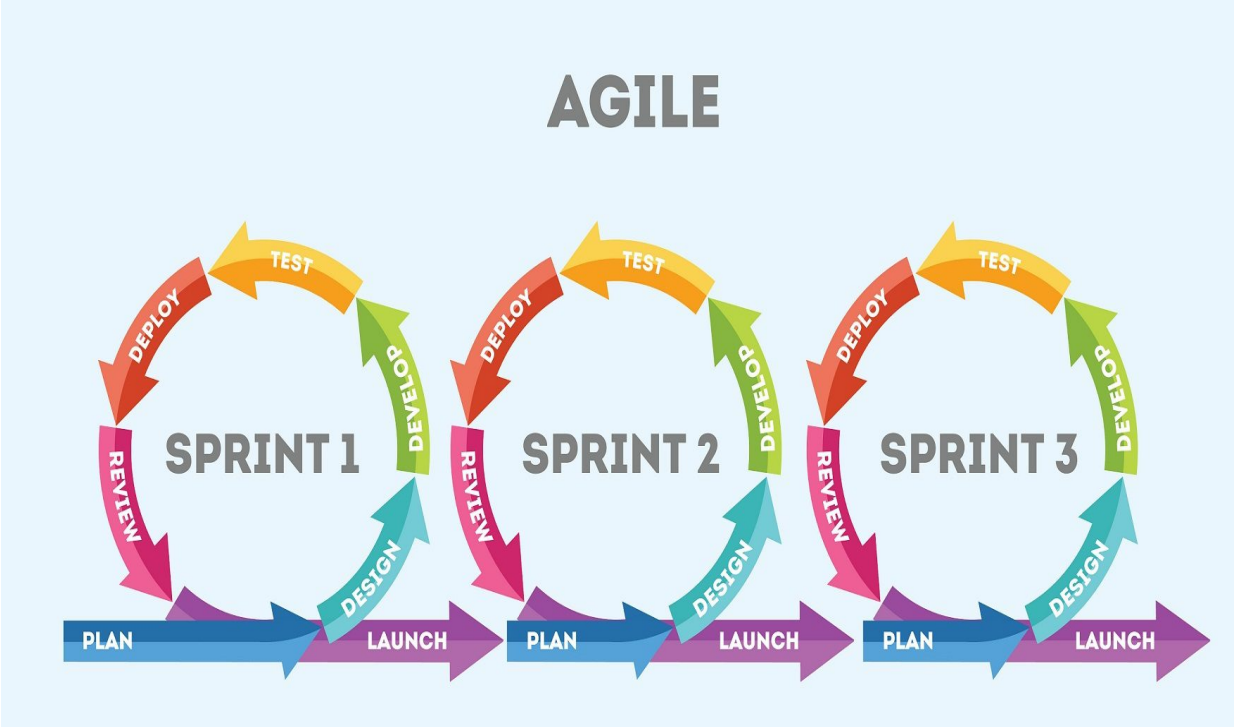


Figure 2 : Agile Methodology

3 - System Analysis

3.1 - Use Case Diagram

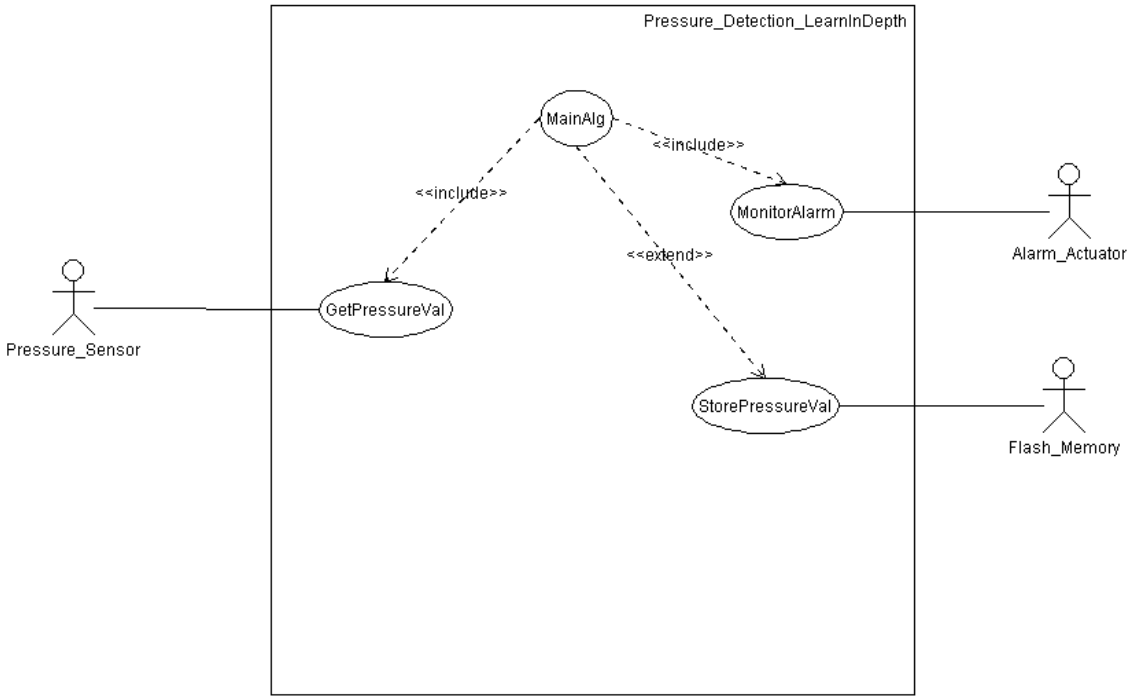


Figure 3 : Use Case Diagram

3.2 - Activity Diagram

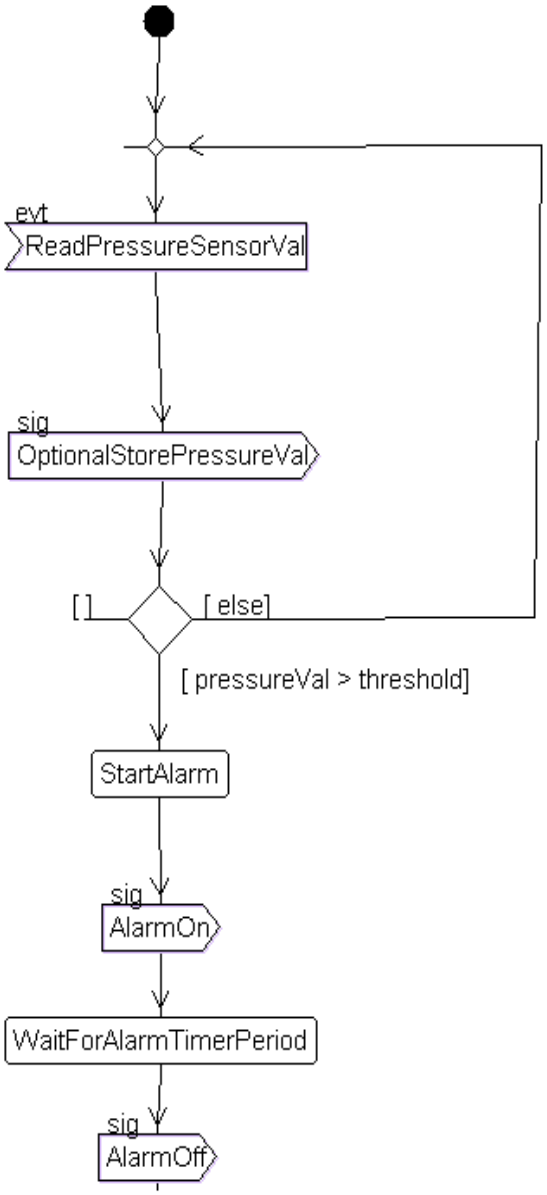


Figure 4 : Activity Diagram

3.3 - Sequence Diagram

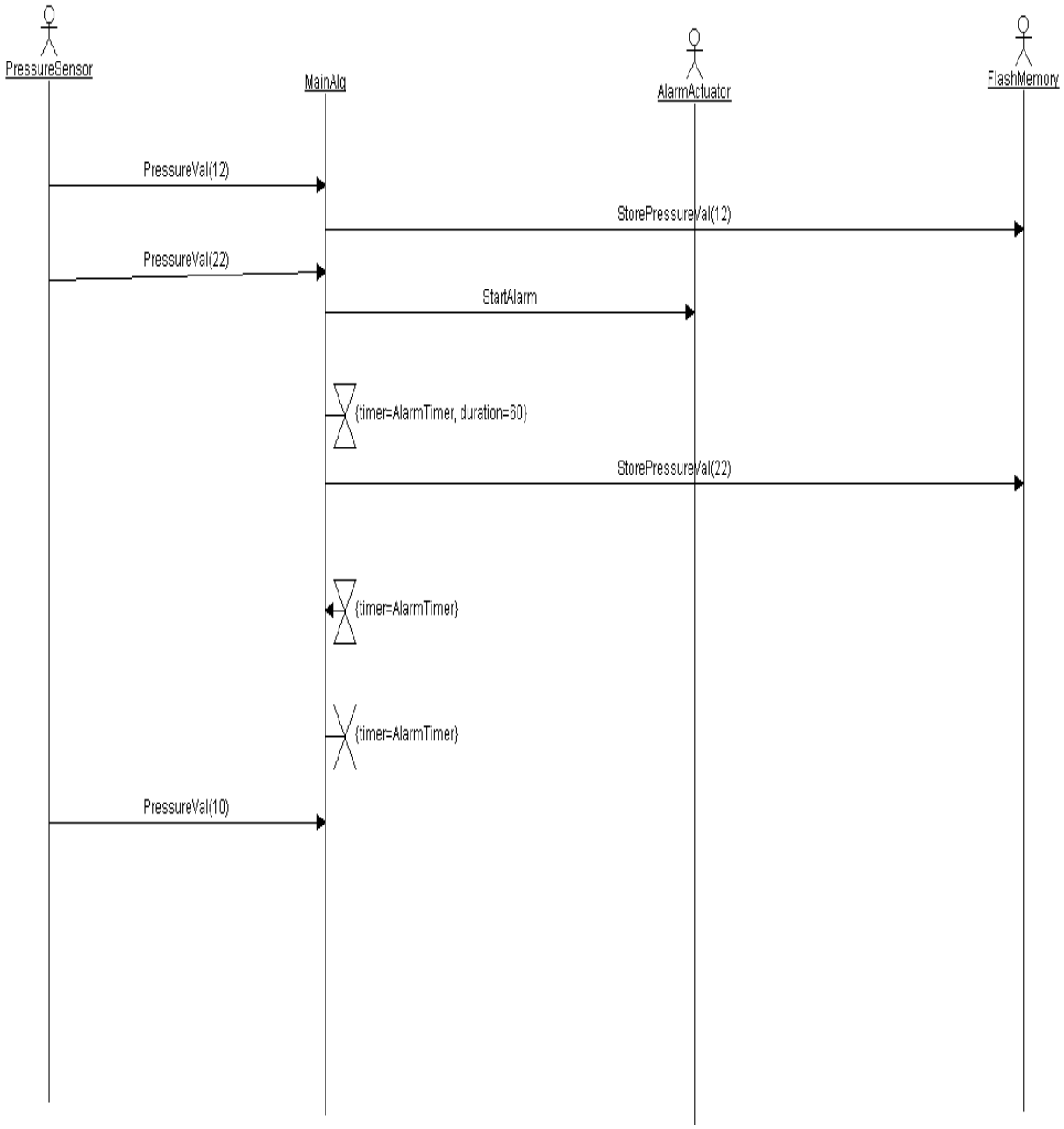


Figure 5 : Sequence Diagram

4 - System Design

4.1 - Block Diagram

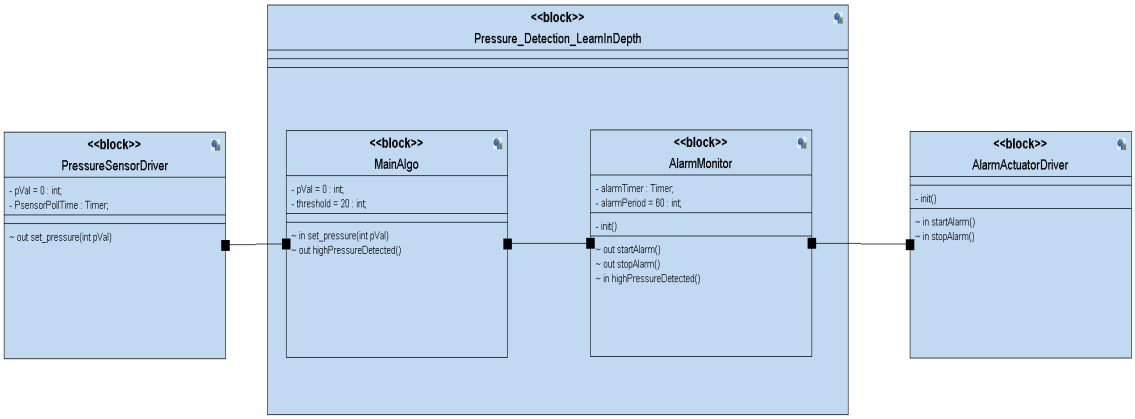


Figure 6 : Block Diagram

4.2 - Pressure Sensor Driver State Machine

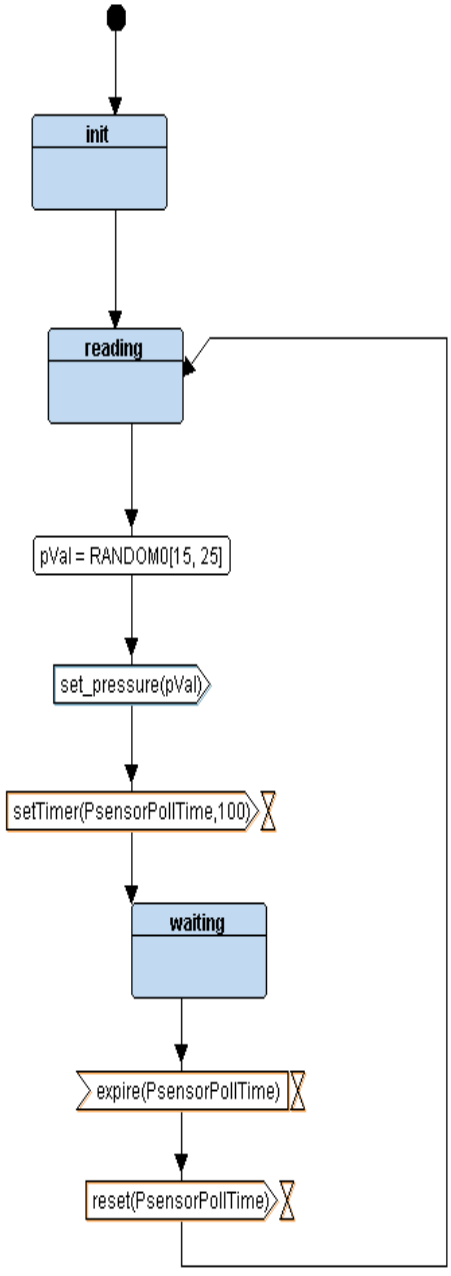


Figure 7 : Pressure Sensor Driver State Machine

4.3 - Alarm Actuator Driver State Machine

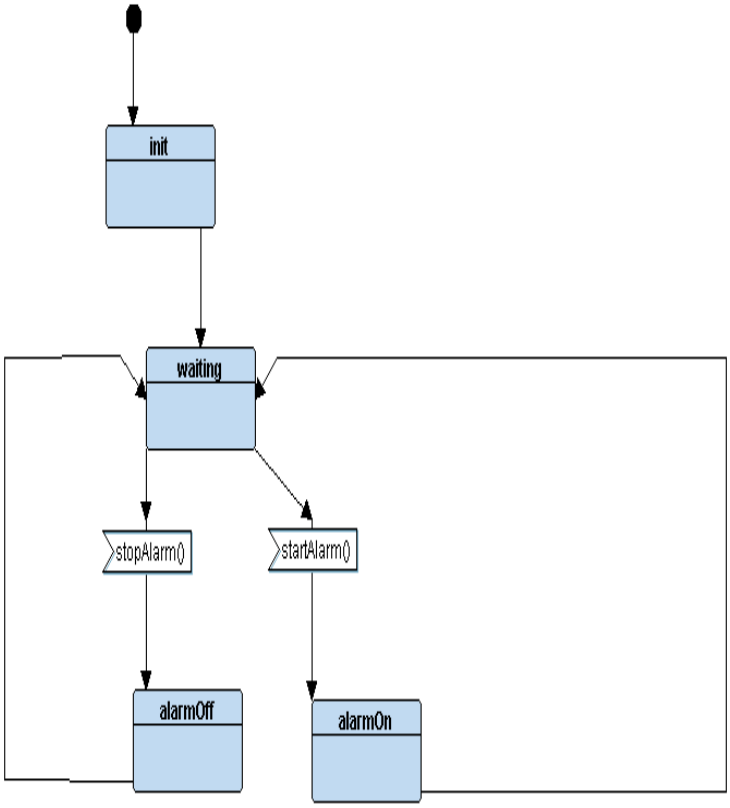


Figure 8 : Alarm Actuator Driver State Machine

4.4 - Alarm Monitor State Machine

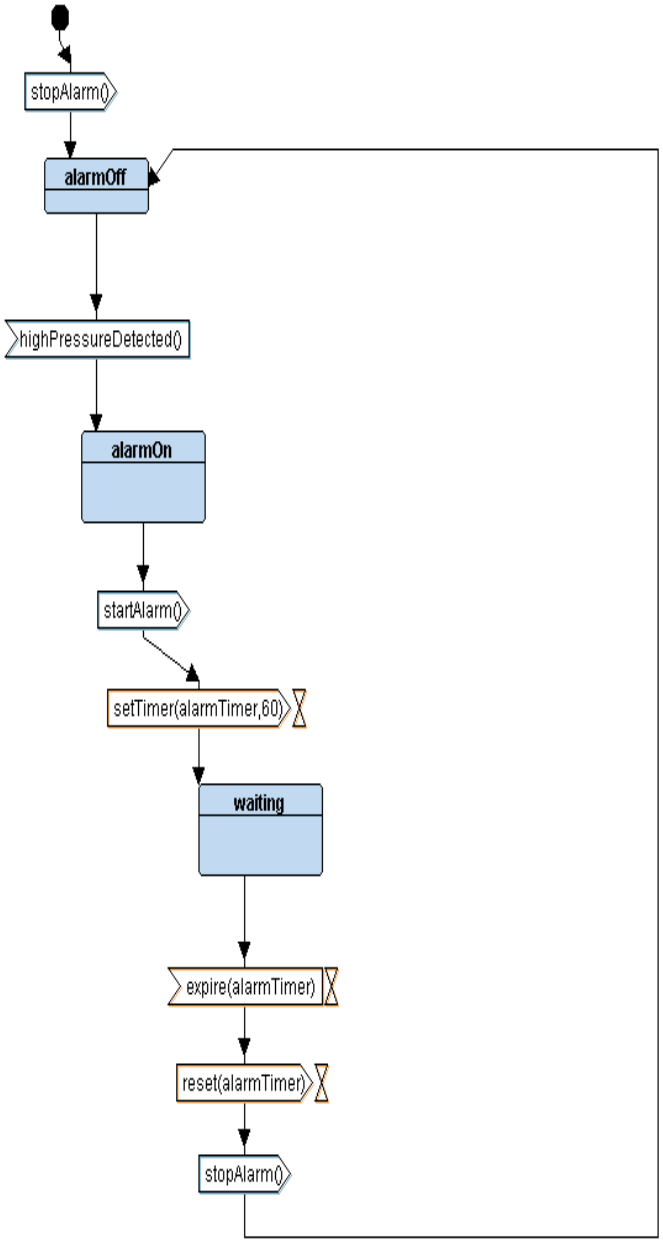


Figure 9 : Alarm Monitor State Machine

4.5 - Main Algorithm State Machine

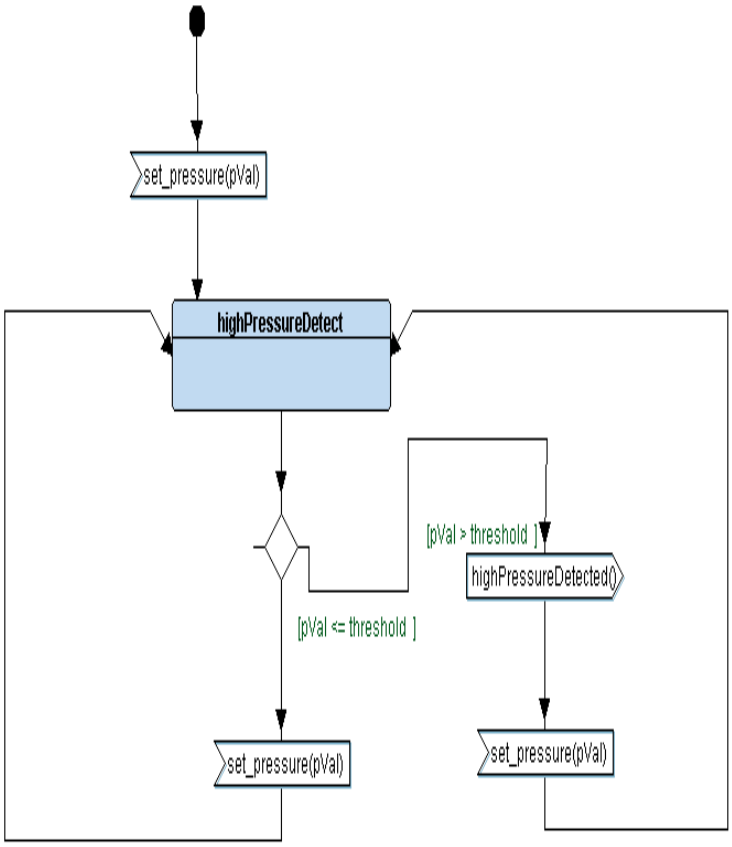


Figure 10 : Main Algorithm State Machine

5 - Source Code

5.1 - Alarm Controller

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

static EN_AC_States_t ACCurrentState;
void (*AC_State)();

void StartAlarm()
{
  ACCurrentState = AC_ALARM_ON; // set current state
  AC_State = STATE(AC_ALARM_ON);  // set the new state
  AC_State();
}
void StopAlarm()
{
  ACCurrentState = AC_ALARM_OFF;  // set current state
  AC_State = STATE(AC_ALARM_OFF); // set the new state
  AC_State();
}

STATE_DEFINE(AC_ALARM_OFF)
{
  Set_Alarm_actuator(AC_ALARM_OFF);    // set the alarm to off
  AC_State = STATE(AC_WAITING); // set new state
}

STATE_DEFINE(AC_ALARM_ON)
{
  Set_Alarm_actuator(AC_ALARM_ON);  // set the alarm to on
  AC_State = STATE(AC_WAITING);   // set new state
}

STATE_DEFINE(AC_WAITING)
{
  ACCurrentState = AC_WAITING;    //set current state
  AC_State = STATE(AC_WAITING);    //set the new state in the pointer
}
```

```c
/*****************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : 00_Pressure_Detection
// File           : AlarmController.h
// Date           : Dec 7, 2023
// GitHub         : https://github.com/sherifkhadr
/*****************************************************************************/

#ifndef ALARMCONTROLLER_H_
#define ALARMCONTROLLER_H_

#include "states.h"


typedef enum
{
    AC_ALARM_ON = 0,
    AC_ALARM_OFF,
    AC_WAITING
}EN_AC_States_t;


STATE_DEFINE(AC_ALARM_OFF);
STATE_DEFINE(AC_ALARM_ON);
STATE_DEFINE(AC_WAITING);

extern void (*AC_State)();

#endif /* ALARMCONTROLLER_H_ */
```

5.2 - Pressure Controller

```c
/*****************************************************************************/
// Author          : Sherif Ashraf Khadr
// Project         : 00_Pressure_Detection
// File            : PressureController.c
// Date            : Dec 7, 2023
// GitHub          : https://github.com/sherifkhadr
/*****************************************************************************/

#include "PressureController.h"
#include "PressureSensor.h"
#include "AlarmController.h"
#include "driver.h"

#define THRESHOLD_VAL        20
static int currentPressure = 0;
void (*PC_State)();

void highPressureDetected()
{
    PC_State = STATE(PC_HIGH_PRESSURE);      // set the state
}

void setPressureVal(int val)
{
    currentPressure = val;   // get the value of the pressure
}

void PC_init(void)
{
    PS_init();
    AC_State = STATE(AC_ALARM_OFF);
    PS_State = STATE(PS_READING);
    PC_State = STATE(PC_BELOW_PRESSURE);
    AC_State();
    PC_State();
}
```

```c
void PC_pressureMonitor(void)
{
    while(1)
    {
        PS_State();
        if(currentPressure >= THRESHOLD_VAL)
            highPressureDetected();
        PC_State();
    }
}

STATE_DEFINE(PC_HIGH_PRESSURE)
{
    StartAlarm();// Start alarm
    Delay(6000); // Delay for 60s
    StopAlarm(); // Stop alarm
    PC_State = STATE(PC_BELOW_PRESSURE); // set the state of sensor module to PC_BELOW_PRESSURE
}

STATE_DEFINE(PC_BELOW_PRESSURE)
{
    /*Do Nothing*/
}
```

```c
/*****************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : 00_Pressure_Detection
// File           : PressureController.h
// Date           : Dec 7, 2023
// GitHub         : https://github.com/sherifkhadr
/*****************************************************************************/


#ifndef PRESSURECONTROLLER_H_
#define PRESSURECONTROLLER_H_

#include "states.h"


typedef enum
{
    PC_CHECK_PRESSURE = 0,
    PC_HIGH_PRESSURE,
    PC_BELOW_PRESSURE
}EN_PC_States_t;

STATE_DEFINE(PC_HIGH_PRESSURE);
STATE_DEFINE(PC_BELOW_PRESSURE);


extern void (*PC_State)();

void PC_init(void);
void PC_pressureMonitor(void);

#endif /* PRESSURECONTROLLER_H_ */
```

5.3 - Pressure Sensor

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

static int pressureVal = 0;
static EN_PS_States_t PSCurrentState;

void (*PS_State)();

void PS_init(void)
{
    GPIO_INITIALIZATION(); // init sensor
    PS_State = STATE(PS_READING);    // set the new state in the pointer
}

STATE_DEFINE(PS_READING)
{
    PSCurrentState = PS_READING;     // set current state
    pressureVal = getPressureVal(); // read the value of the sensor
    PS_State = STATE(PS_SEND_VAL);  // set the new state in the pointer
}

STATE_DEFINE(PS_SEND_VAL)
{
    PSCurrentState = PS_SEND_VAL;    // set current state
    setPressureVal(pressureVal);     // update the value
    PS_State = STATE(PS_WAITING);    // set the new state in the pointer
}

STATE_DEFINE(PS_WAITING)
{
    PSCurrentState = PS_WAITING;     // set current state
    Delay(10000);    // wait for the delay
    PS_State = STATE(PS_READING);    // set the new state in the pointer
}
```

```c
/*****************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : 00_Pressure_Detection
// File           : PressureSensor.h
// Date           : Dec 7, 2023
// GitHub         : https://github.com/sherifkhadr
/*****************************************************************************/

#ifndef PRESSURESENSOR_H_
#define PRESSURESENSOR_H_

#include "states.h"


typedef enum
{
    PS_READING = 0,
    PS_SEND_VAL,
    PS_WAITING
}EN_PS_States_t;

void PS_init(void);

STATE_DEFINE(PS_READING);
STATE_DEFINE(PS_SEND_VAL);
STATE_DEFINE(PS_WAITING);

extern void (*PS_State)();

#endif /* PRESSURESENSOR_H_ */
```

5.4 - Main

```c
/*****************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : 00_Pressure_Detection
// File           : main.c
// Date           : Dec 7, 2023
// GitHub         : https://github.com/sherifkhadr
/*****************************************************************************/


#include "PressureController.h"


int main(void)
{

    PC_init();
    PC_pressureMonitor();
    return 0;
}
```

5.5 - Driver

```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_Alarm_actuator(int i){
    if (i == 1){
        SET_BIT(GPIOA_ODR,13);
    }
    else if (i == 0){
        RESET_BIT(GPIOA_ODR,13);
    }
}

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

```

```c
#include <stdint.h>
#include <stdio.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_Alarm_actuator(int i);
void GPIO_INITIALIZATION ();
```

5.6 - States

```c
/*****************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : 00_Pressure_Detection
// File           : states.h
// Date           : Dec 7, 2023
// GitHub         : https://github.com/sherifkhadr
/*****************************************************************************/

#ifndef STATES_H_
#define STATES_H_

#include <stdio.h>
#include <stdlib.h>

//Automatic STATE Function genrated
#define STATE_DEFINE(_stateFun_) void ST_##_stateFun_()
#define STATE(_stateFun_)   ST_##_stateFun_

// Signals
void setPressureVal(int val);
void highPressureDetected();
void StartAlarm();
void StopAlarm();

#endif /* STATES_H_ */
```

5.7 - Makefile

```
#/*****************************************************************************/
#// Author         : Sherif Ashraf Khadr
#// Project        : 00_Pressure_Detection
#// File           : Makefile
#// Date           : Dec 7, 2023
#// GitHub         : https://github.com/sherifkhadr
#/*****************************************************************************/


CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -gdwarf-2
INCS=-I .
LIBS=
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
AS = $(wildcard *.s)
ASOBJ = $(AS:.s=.o)
projectName=Pressure_Detection

all: $(projectName).bin
    @echo "Build Is Done"

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

$(projectName).elf: $(OBJ) $(ASOBJ)
    $(CC)ld.exe -T linkerscript.ld $(LIBS) $(OBJ) $(ASOBJ) -o $(projectName).elf -Map=$(projectName).map

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

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

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

5.8 - Startup File

```c
#include <stdint.h>

extern int main(void);
extern unsigned int _E_TEXT;
extern unsigned int _S_DATA;
extern unsigned int _E_DATA;
extern unsigned int _S_BSS;
extern unsigned int _E_BSS;
extern unsigned int _STACK_TOP;

void Default_Handler(void);
void Reset_Handler(void);
void NMI_Handler(void) __attribute__((weak,alias("Default_Handler")));;
void H_Fault_Handler(void) __attribute__((weak,alias("Default_Handler")));;
void MM_Fault_Handler(void) __attribute__((weak,alias("Default_Handler")));;
void Bus_Fault_Handler(void) __attribute__((weak,alias("Default_Handler")));;
void Usage_Fault_Handler(void) __attribute__((weak,alias("Default_Handler")));;

uint32_t vectors[] __attribute__((section(".vectors"))) =
{
    (uint32_t) &_STACK_TOP,
    (uint32_t) &Reset_Handler,
    (uint32_t) &NMI_Handler,
    (uint32_t) &H_Fault_Handler,
    (uint32_t) &MM_Fault_Handler,
    (uint32_t) &Bus_Fault_Handler,
    (uint32_t) &Usage_Fault_Handler,

};

void Default_Handler(void)
{
        Reset_Handler();
}
```

```c
void Reset_Handler(void)
{
    /* copy data section from flash to sram */

    unsigned int DATA_SIZE = (unsigned char *)&_E_DATA - (unsigned char *)&_S_DATA ;
    unsigned char*  P_src = (unsigned char *)&_E_TEXT ;
    unsigned char*  P_dst = (unsigned char *)&_S_DATA ;

    for(int i=0 ; i<DATA_SIZE ; i++)
    {
        *((unsigned char *)P_dst++) = *((unsigned char *)P_src++);
    }

    /* init .bss section in sram  = 0*/

    unsigned int BSS_SIZE = (unsigned char *)&_E_BSS - (unsigned char *)&_S_BSS ;
    P_dst = (unsigned char*)&_S_BSS;

    for(int i=0 ; i<BSS_SIZE ; i++)
    {
        *((unsigned char *)P_dst++) = (unsigned char)0;
    }

    /* jump to the main */

    main();
}
```

5.9 - Linker Script File

```
10  MEMORY
11  {
12          flash(RX) : ORIGIN = 0x08000000 , LENGTH = 128K
13          sram(RWX) : ORIGIN = 0x20000000 , LENGTH = 20K
14  }
15
16  SECTIONS
17  {
18
19      .text : {
20                      *(.vectors*)
21                      *(.text*)
22                      *(.rodata*)
23                      _E_TEXT = . ;
24      }> flash
25
26      .data : {
27                  _S_DATA = . ;
28                  *(.data*)
29                  . = ALIGN(4) ;
30                  _E_DATA = . ;
31      }> sram AT> flash
32
33      .bss : {
34                  _S_BSS = . ;
35                  *(.bss*)
36                  _E_BSS = . ;
37                  . = ALIGN(4) ;
38                  . = . + 0x1000 ;
39                  _STACK_TOP = . ;
40      }> sram
41  }
```

5.10 - Symbols File

```
arm-none-eabi-nm.exe .\Pressure_Detection.elf

2000000c B _E_BSS
20000000 D _E_DATA
080003b0 T _E_TEXT
20000000 B _S_BSS
20000000 D _S_DATA
2000100c B _STACK_TOP
20001010 B AC_State
20000005 b ACCurrentState
0800001c W Bus_Fault_Handler
20000008 b currentPressure
0800001c T Default_Handler
080002ec T Delay
0800030c T getPressureVal
08000360 T GPIO_INITIALIZATION
0800001c W H_Fault_Handler
08000210 T highPressureDetected
080000ac T main
0800001c W MM_Fault_Handler
0800001c W NMI_Handler
08000248 T PC_init
0800028c T PC_pressureMonitor
20001014 B PC_State
20000000 b pressureVal
080000c0 T PS_init
2000100c B PS_State
20000004 b PSCurrentState
08000028 T Reset_Handler
08000324 T Set_Alarm_actuator
0800022c T setPressureVal
080001b4 T ST_AC_ALARM_OFF
080001d0 T ST_AC_ALARM_ON
080001ec T ST_AC_WAITING
080002e0 T ST_PC_BELOW_PRESSURE
080002b8 T ST_PC_HIGH_PRESSURE
080000dc T ST_PS_READING
0800010c T ST_PS_SEND_VAL
0800013c T ST_PS_WAITING
08000164 T StartAlarm
0800018c T StopAlarm
0800001c W Usage_Fault_Handler
08000000 T vectors
```

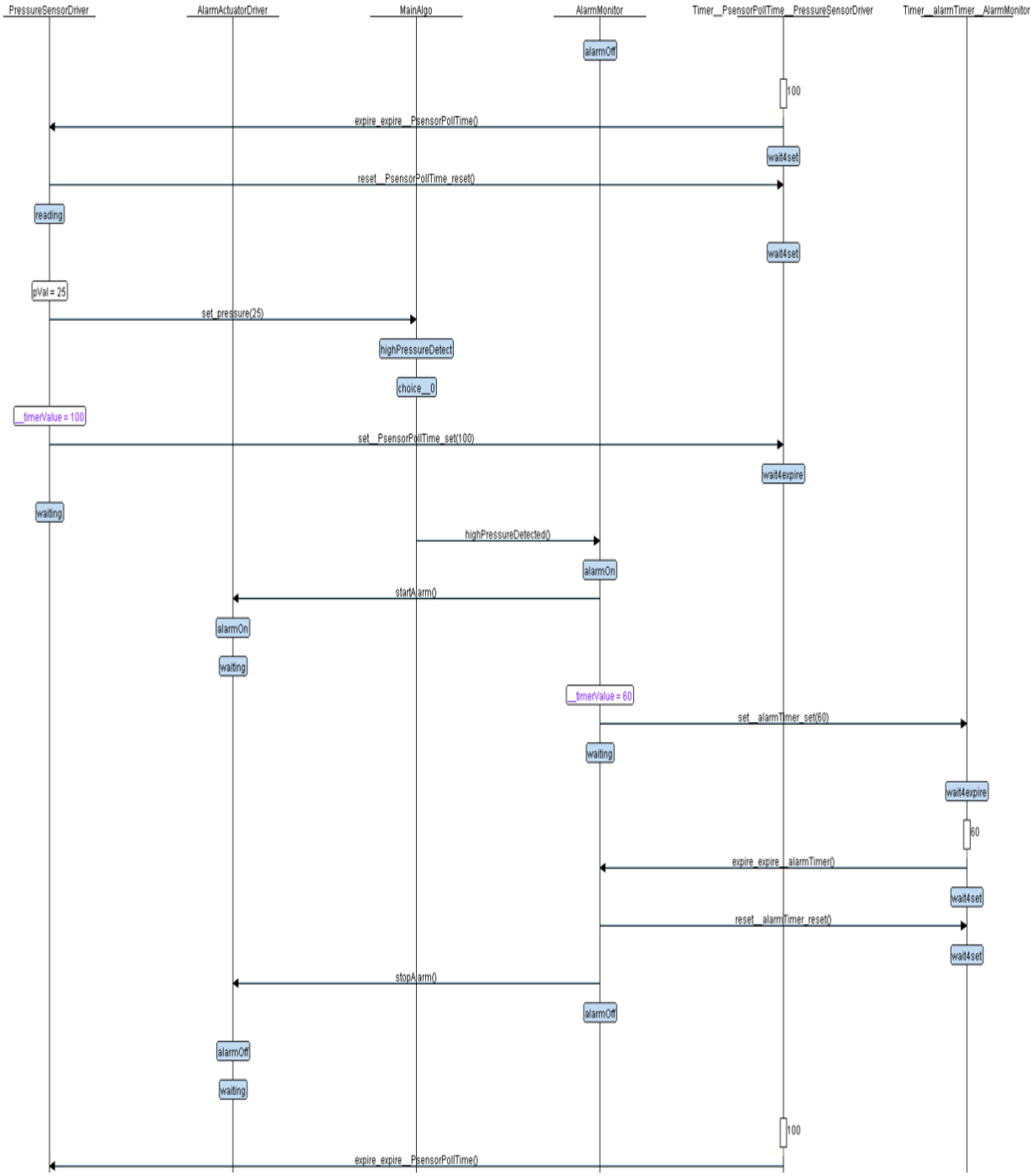
5.11 - Map File

```
s
Allocating common symbols
Common symbol        size              file

PC_State             0x4               PressureController.o
PS_State             0x4               PressureSensor.o
AC_State             0x4               AlarmController.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      0x3b0
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x0800001c       0x90 startup.o
                0x0800001c                Bus_Fault_Handler
                0x0800001c                H_Fault_Handler
                0x0800001c                MM_Fault_Handler
                0x0800001c                Usage_Fault_Handler
                0x0800001c                Default_Handler
                0x0800001c                NMI_Handler
                0x08000028                Reset_Handler
 .text          0x080000ac       0x12 main.o
                0x080000ac                main
 *fill*         0x080000be        0x2
 .text          0x080000c0       0xa4 PressureSensor.o
                0x080000c0                PS_init
                0x080000dc                ST_PS_READING
                0x0800010c                ST_PS_SEND_VAL
                0x0800013c                ST_PS_WAITING
 .text          0x08000164       0xac AlarmController.o
                0x08000164                StartAlarm
                0x0800018c                StopAlarm
                0x080001b4                ST_AC_ALARM_OFF
                0x080001d0                ST_AC_ALARM_ON
                0x080001ec                ST_AC_WAITING
 .text          0x08000210       0xdc PressureController.o
                0x08000210                highPressureDetected
                0x0800022c                setPressureVal
                0x08000248                PC_init
                0x0800028c                PC_pressureMonitor
                0x080002b8                ST_PC_HIGH_PRESSURE
                0x080002e0                ST_PC_BELOW_PRESSURE
 .text          0x080002ec       0xc4 driver.o
                0x080002ec                Delay
                0x0800030c                getPressureVal
                0x08000324                Set_Alarm_actuator
                0x08000360                GPIO_INITIALIZATION
 *(.rodata*)
                0x080003b0                _E_TEXT = .

.glue_7         0x080003b0        0x0
 .glue_7        0x080003b0        0x0 linker stubs

.glue_7t        0x080003b0        0x0
 .glue_7t       0x080003b0        0x0 linker stubs

.vfp11_veneer   0x080003b0        0x0
 .vfp11_veneer  0x080003b0        0x0 linker stubs

.v4_bx          0x080003b0        0x0
 .v4_bx         0x080003b0        0x0 linker stubs

.iplt           0x080003b0        0x0
```

```
.iplt           0x080003b0        0x0
 .iplt          0x080003b0        0x0 startup.o

.rel.dyn        0x080003b0        0x0
 .rel.iplt      0x080003b0        0x0 startup.o

.data           0x20000000        0x0 load address 0x080003b0
                0x20000000                _S_DATA = .
 *(.data*)
 .data          0x20000000        0x0 startup.o
 .data          0x20000000        0x0 main.o
 .data          0x20000000        0x0 PressureSensor.o
 .data          0x20000000        0x0 AlarmController.o
 .data          0x20000000        0x0 PressureController.o
 .data          0x20000000        0x0 driver.o
                0x20000000                . = ALIGN (0x4)
                0x20000000                _E_DATA = .

.igot.plt       0x20000000        0x0 load address 0x080003b0
 .igot.plt      0x20000000        0x0 startup.o

.bss            0x20000000     0x1018 load address 0x080003b0
                0x20000000                _S_BSS = .
 *(.bss*)
 .bss           0x20000000        0x0 startup.o
 .bss           0x20000000        0x0 main.o
 .bss           0x20000000        0x5 PressureSensor.o
 .bss           0x20000005        0x1 AlarmController.o
 *fill*         0x20000006        0x2
 .bss           0x20000008        0x4 PressureController.o
 .bss           0x2000000c        0x0 driver.o
                0x2000000c                _E_BSS = .
                0x2000000c                . = ALIGN (0x4)
                0x2000100c                . = (. + 0x1000)
 *fill*         0x2000000c     0x1000
                0x2000100c                _STACK_TOP = .
 COMMON         0x2000100c        0x4 PressureSensor.o
                0x2000100c                PS_State
 COMMON         0x20001010        0x4 AlarmController.o
                0x20001010                AC_State
 COMMON         0x20001014        0x4 PressureController.o
                0x20001014                PC_State
LOAD startup.o
LOAD main.o
LOAD PressureSensor.o
LOAD AlarmController.o
LOAD PressureController.o
LOAD driver.o
OUTPUT(Pressure_Detection.elf elf32-littlearm)
```

6 - Simulation Results

6.1 - Ttool Result



6.2 - Proteus Result