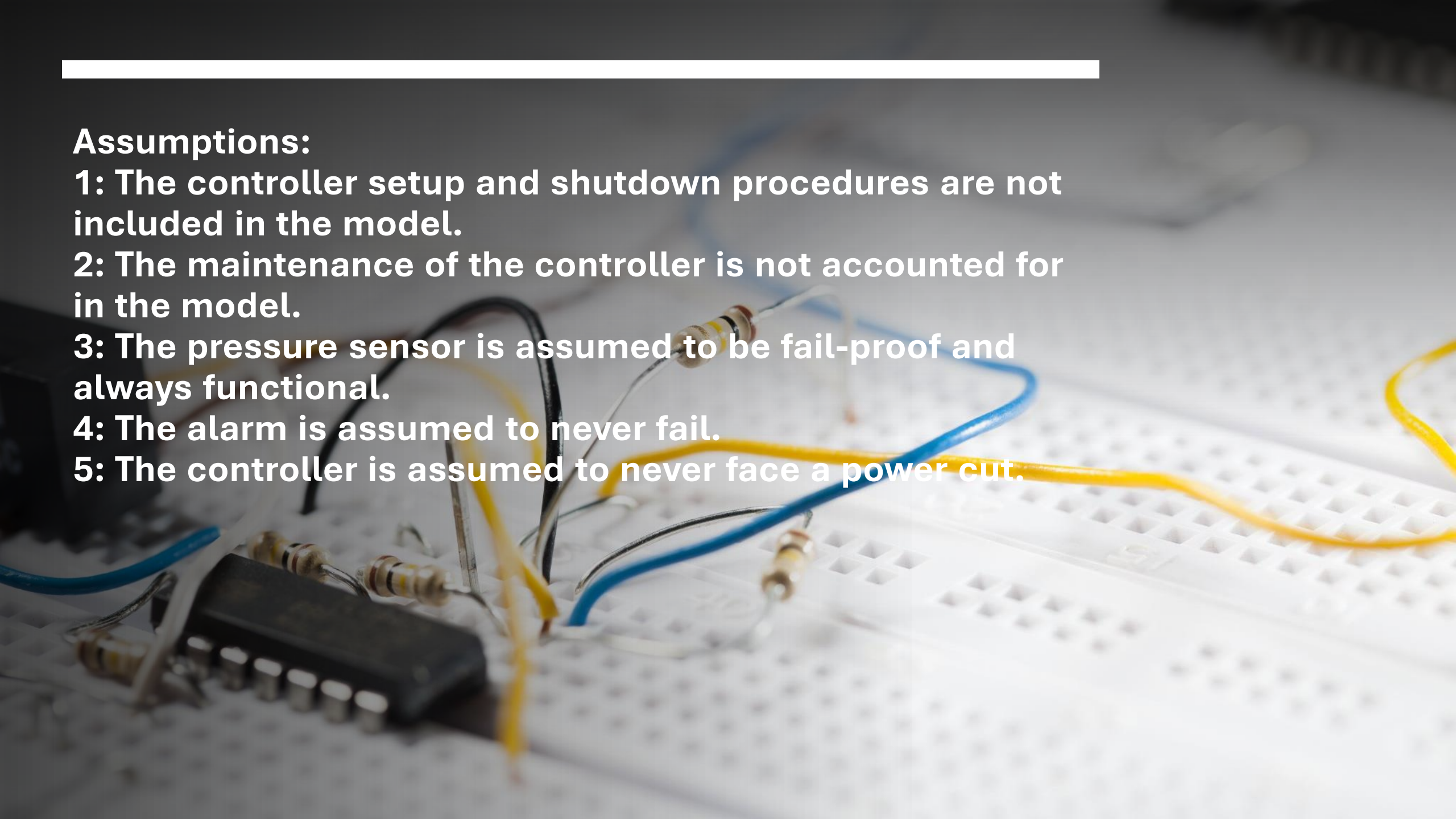
Pressure Alarm Tracer

First Term Final Project

Introduction:

This system focuses on pressure detection in cabin environments. It ensures safety by activating an alarm if cabin pressure exceeds 20 bars. The alarm operates for a duration of 60 seconds, providing adequate time for crew response. Additionally, an optional feature allows tracking of measured pressure values for monitoring and analysis.



Assumptions:

1: The controller setup and shutdown procedures are not included in the model.

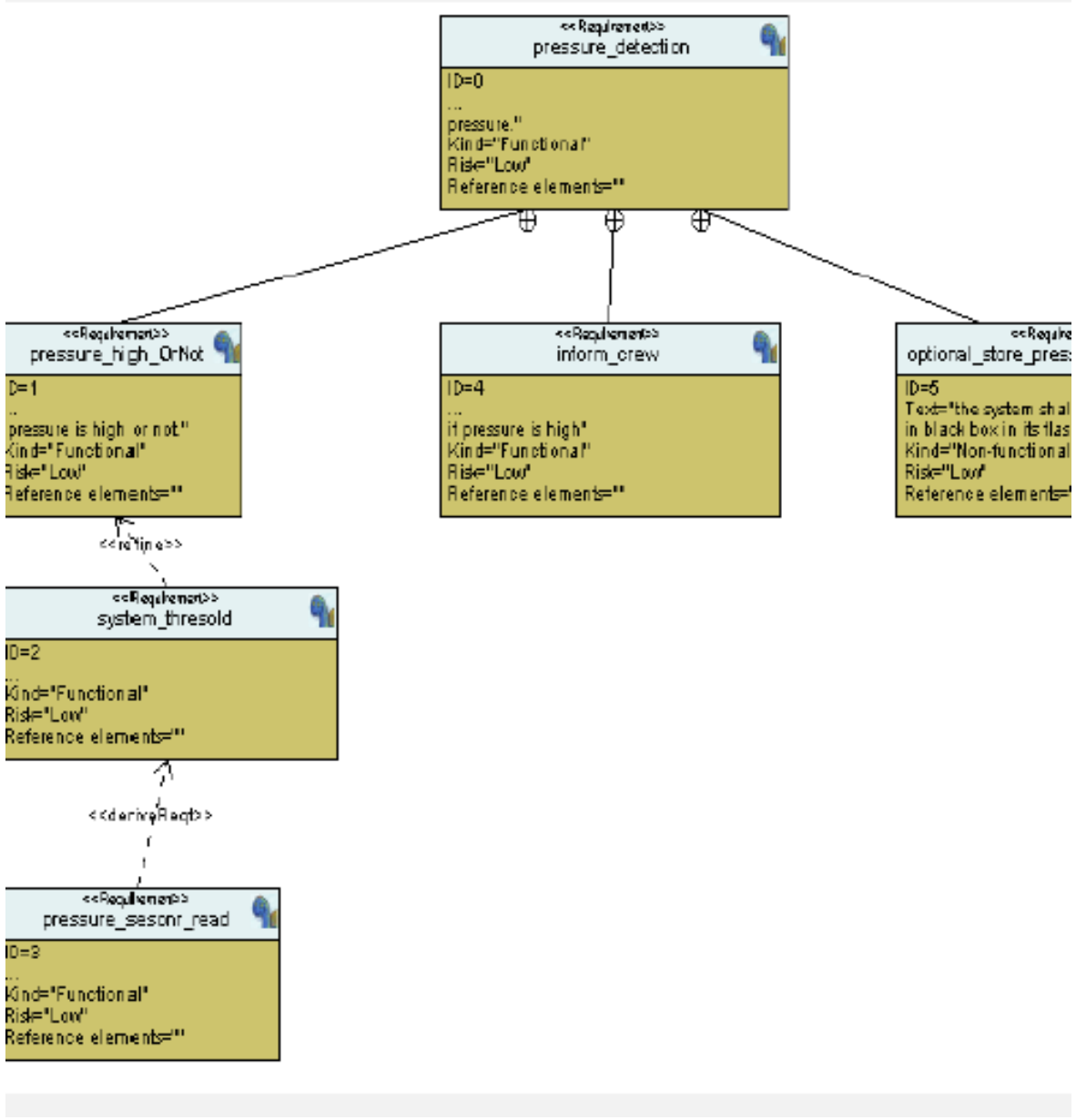
2: The maintenance of the controller is not accounted for in the model.

3: The pressure sensor is assumed to be fail-proof and always functional.

4: The alarm is assumed to never fail.

5: The controller is assumed to never face a power cut.

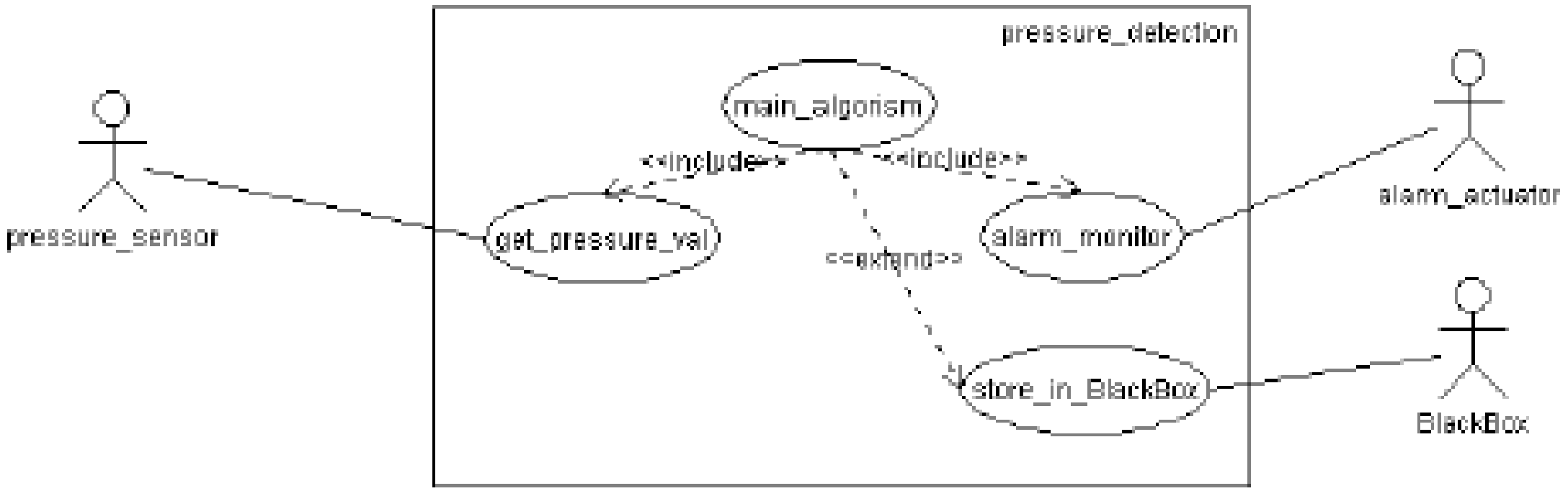
Requirnment



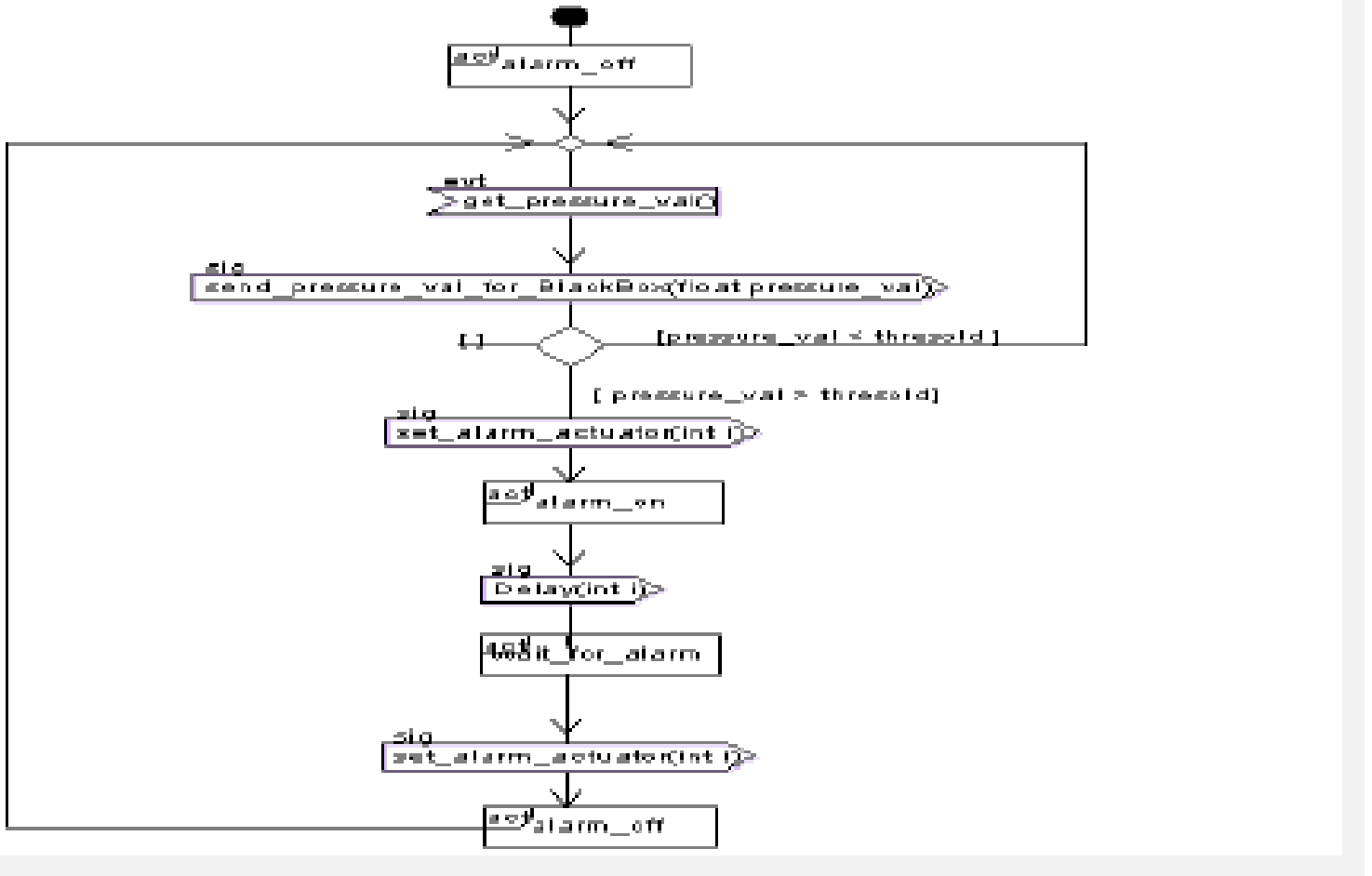
Requirement Digram

System anaylsis

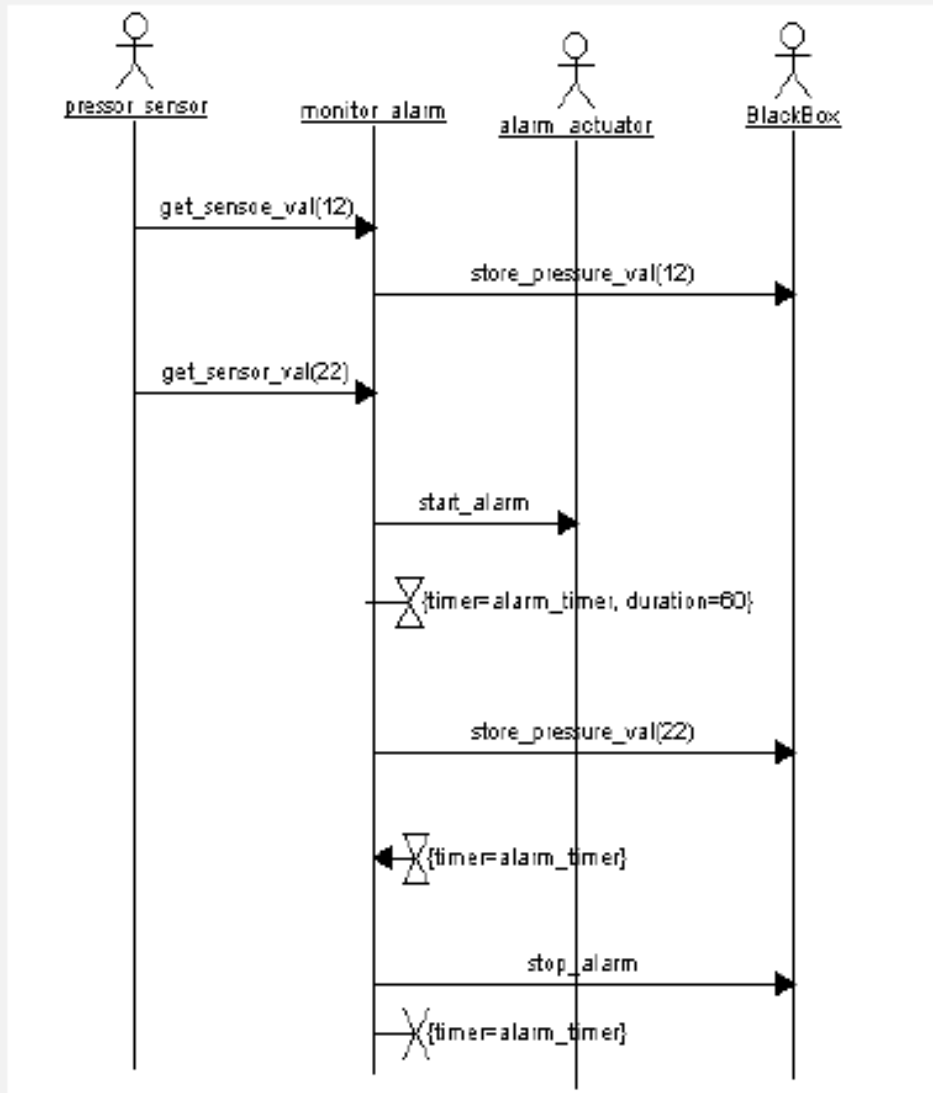
Use Case Gigram



Activity Digram

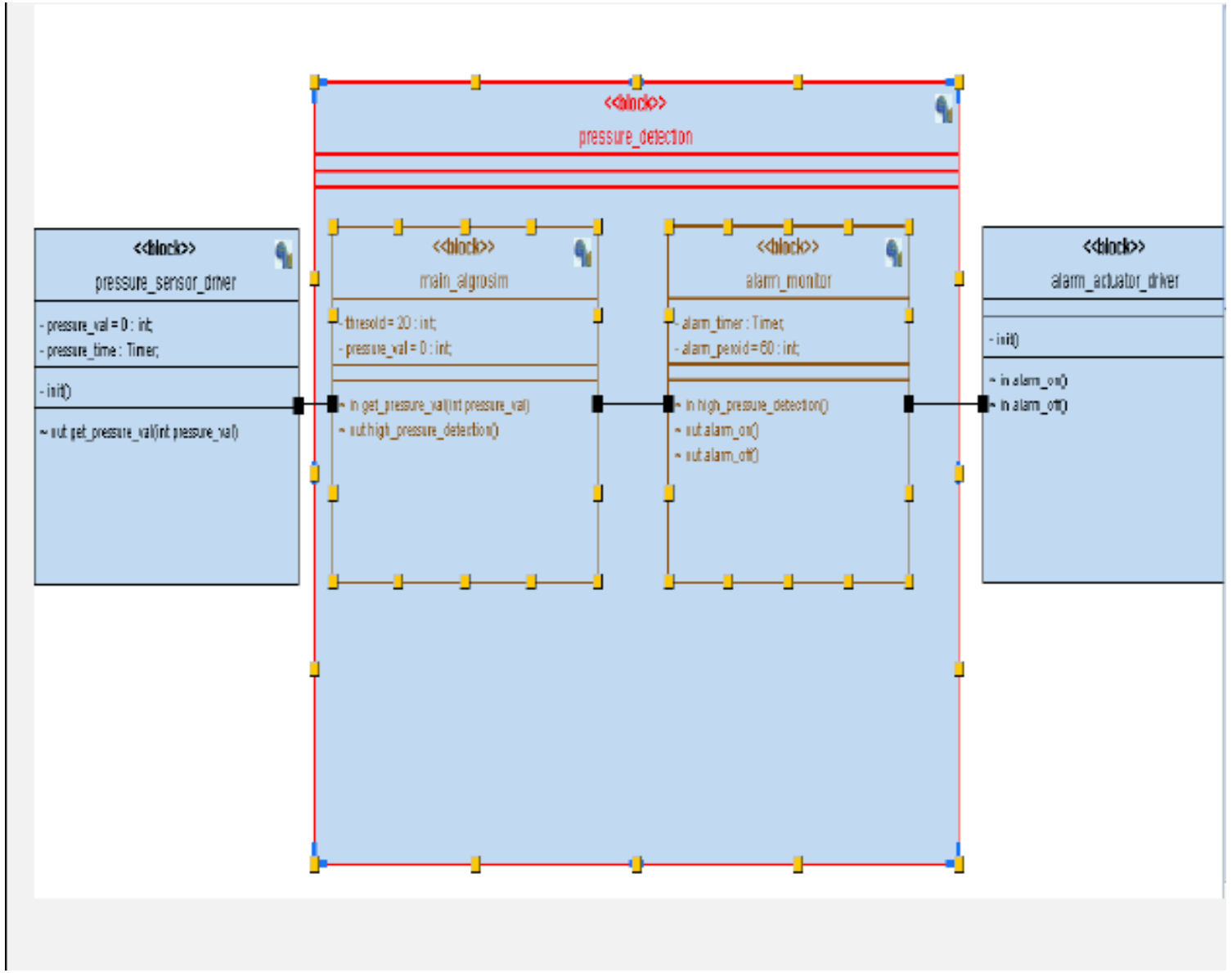


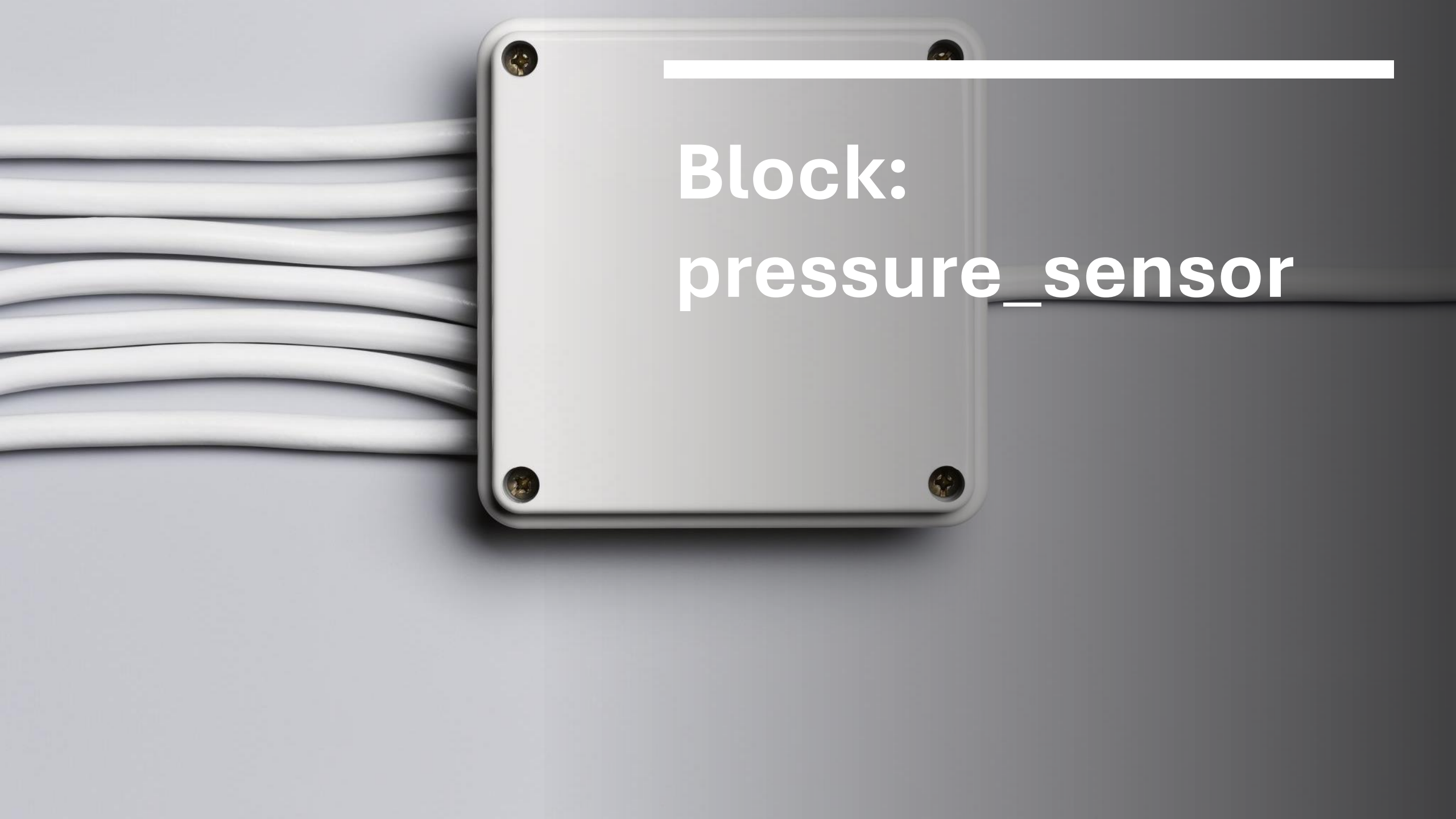
Sequence Digram



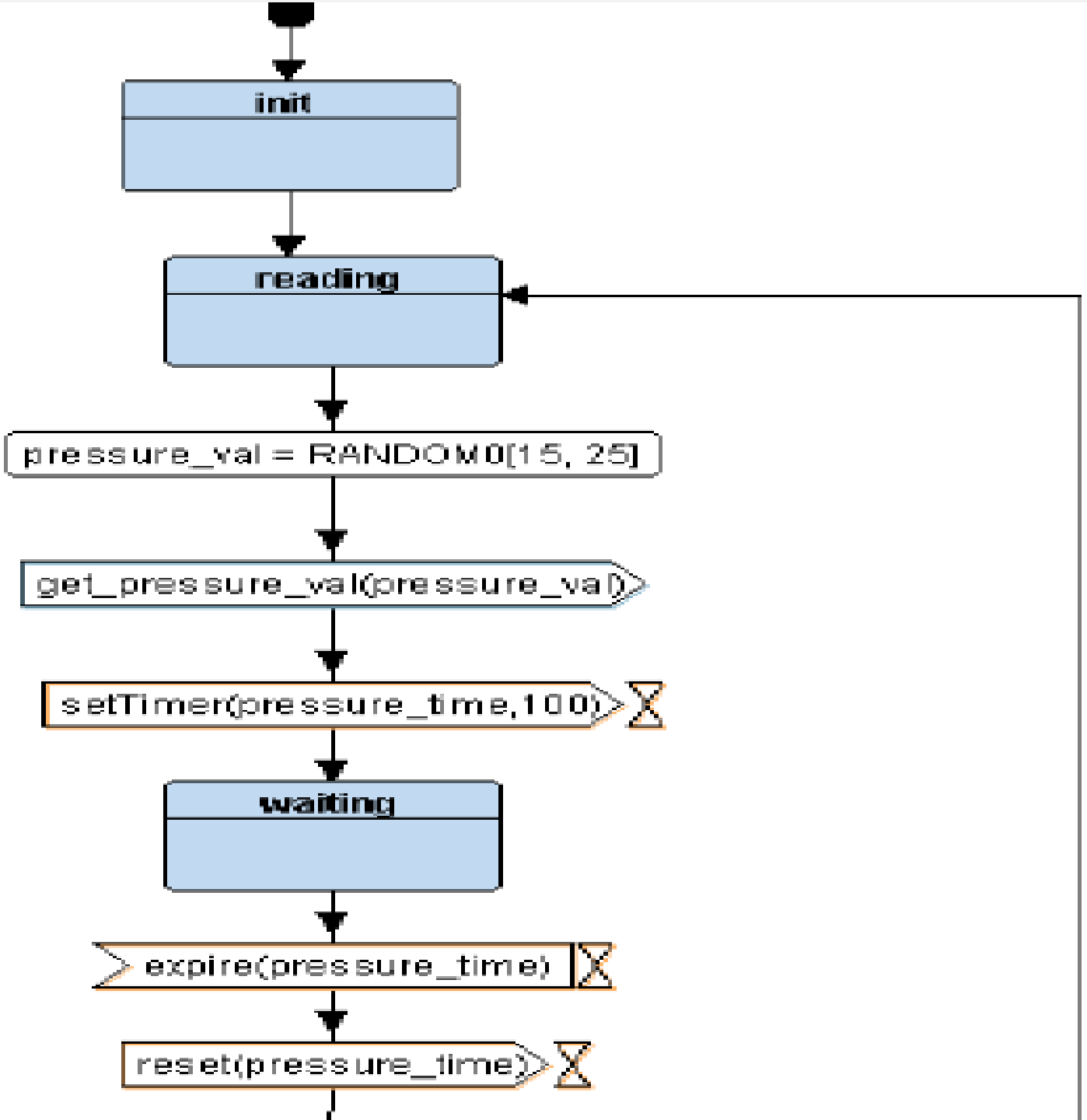
System design

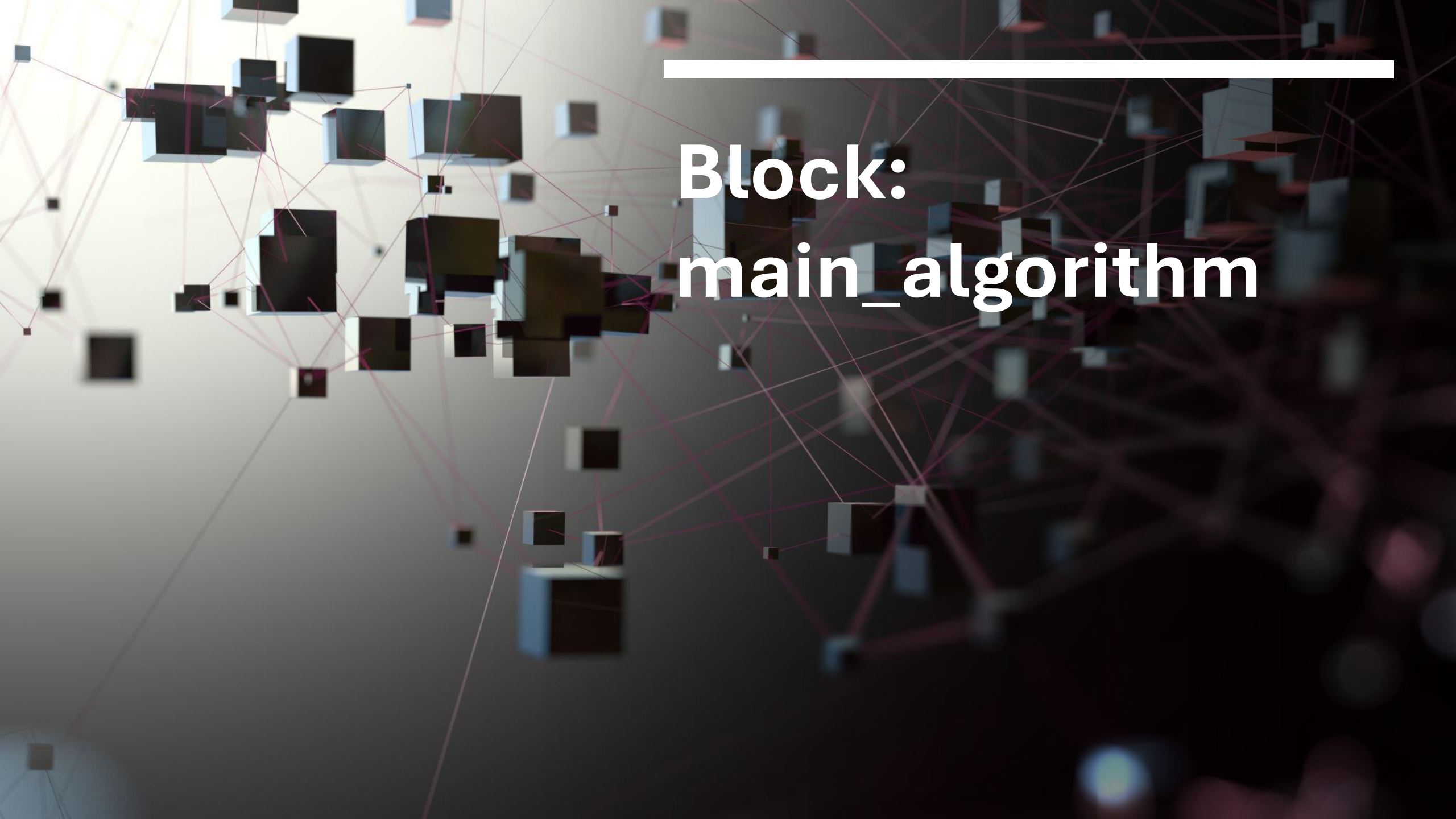
Blocks Digram



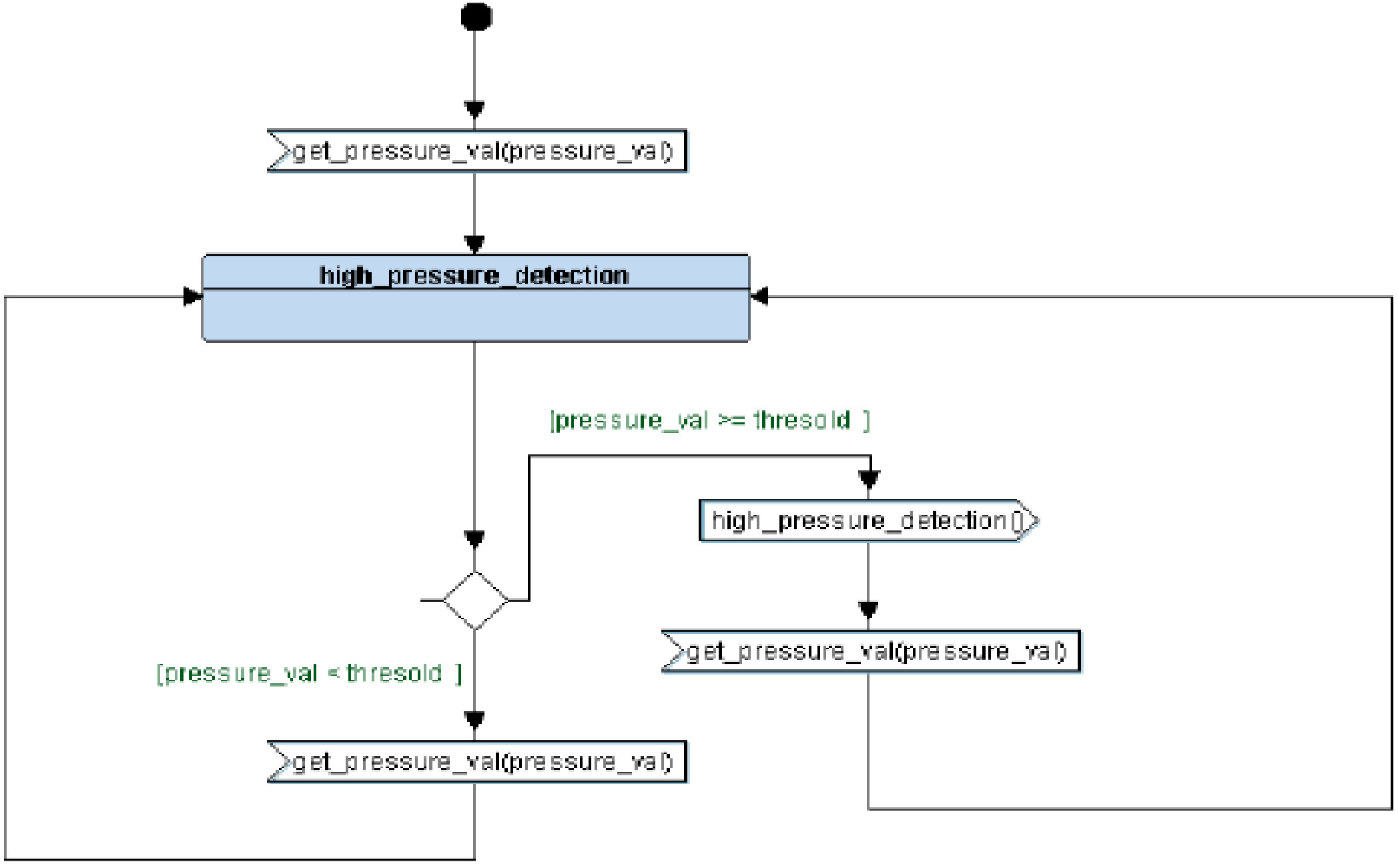


Block: pressure_sensor

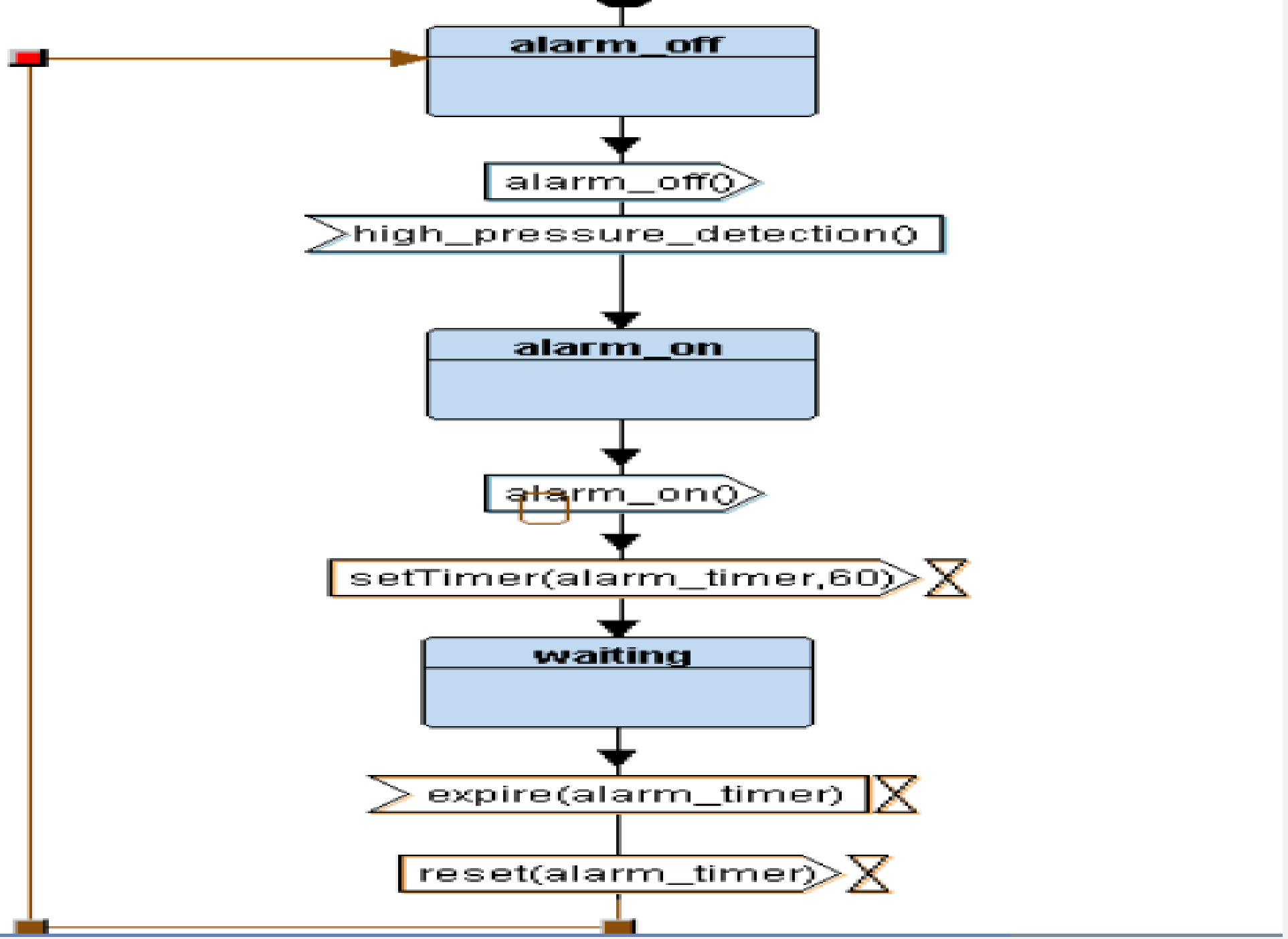

init
reading
pressure_val = RANDOM0[15, 25]
get_pressure_val(pressure_val)
setTimer(pressure_time,100)
waiting
expire(pressure_time)
reset(pressure_time)



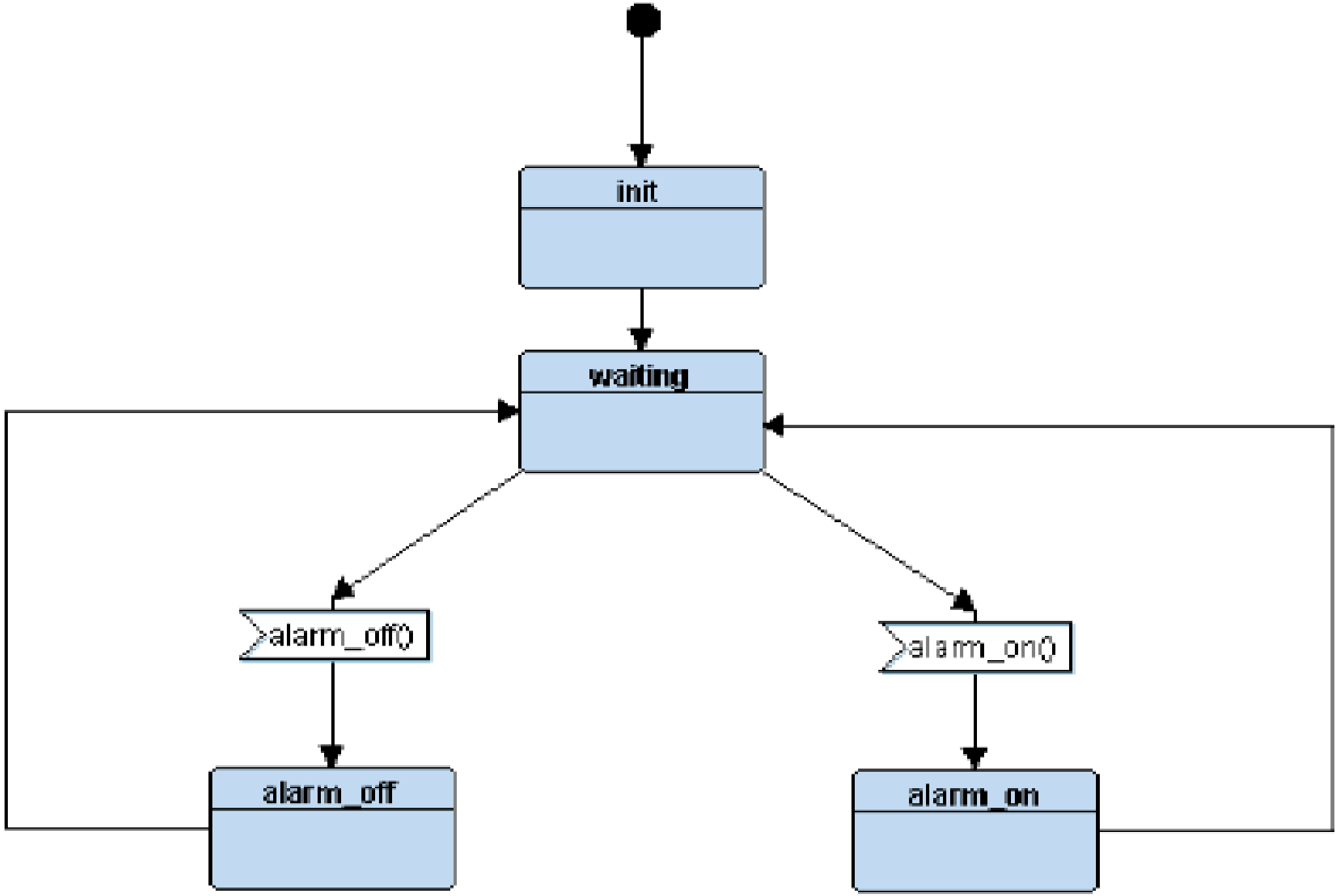
Block: main_algorithm

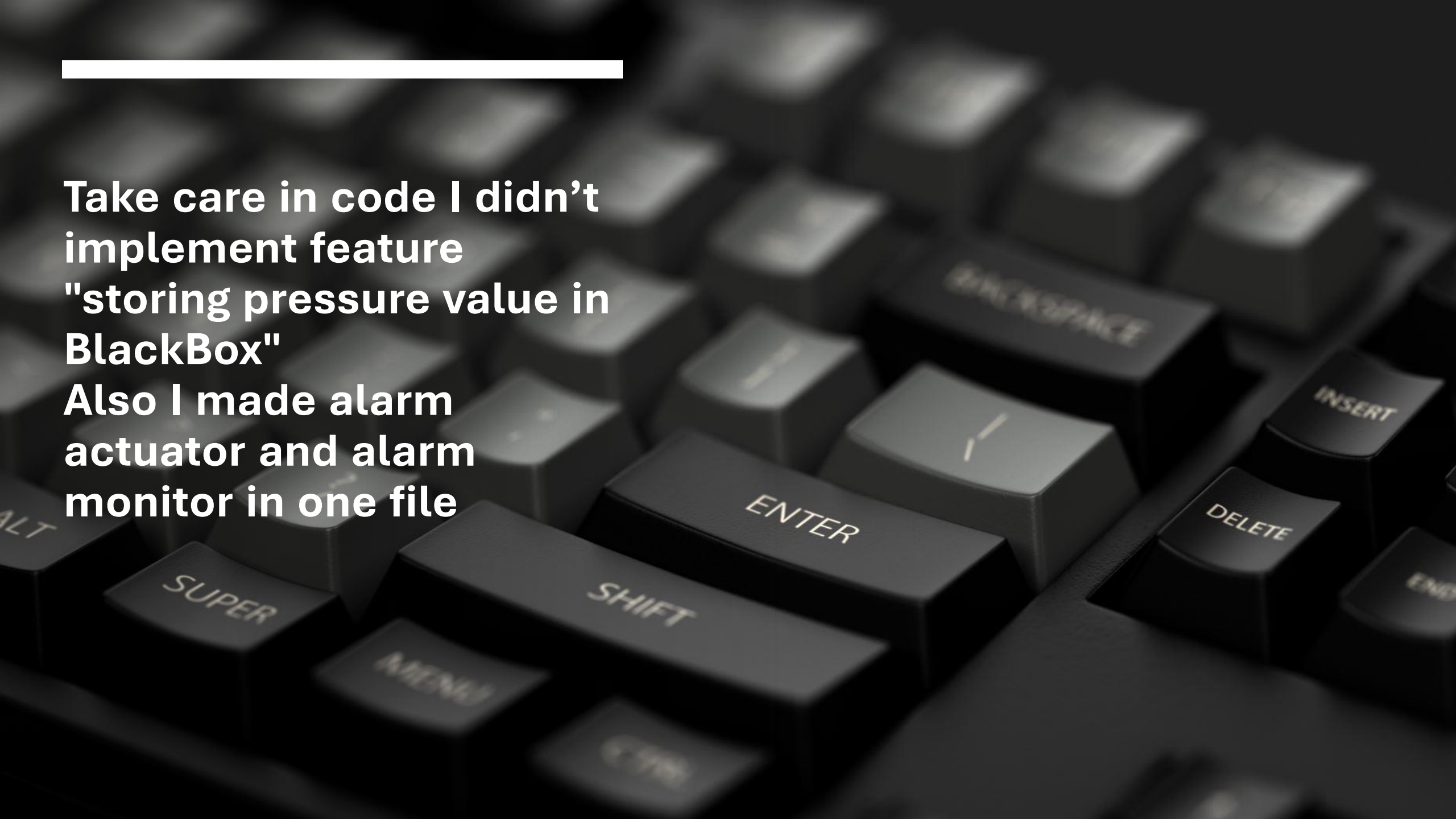

get_pressure_val(pressure_val)
high_pressure_detection
[pressure_val >= thresold]
high_pressure_detection()
get_pressure_val(pressure_val)
[pressure_val < thresold]
get_pressure_val(pressure_val)

alarm_monitor

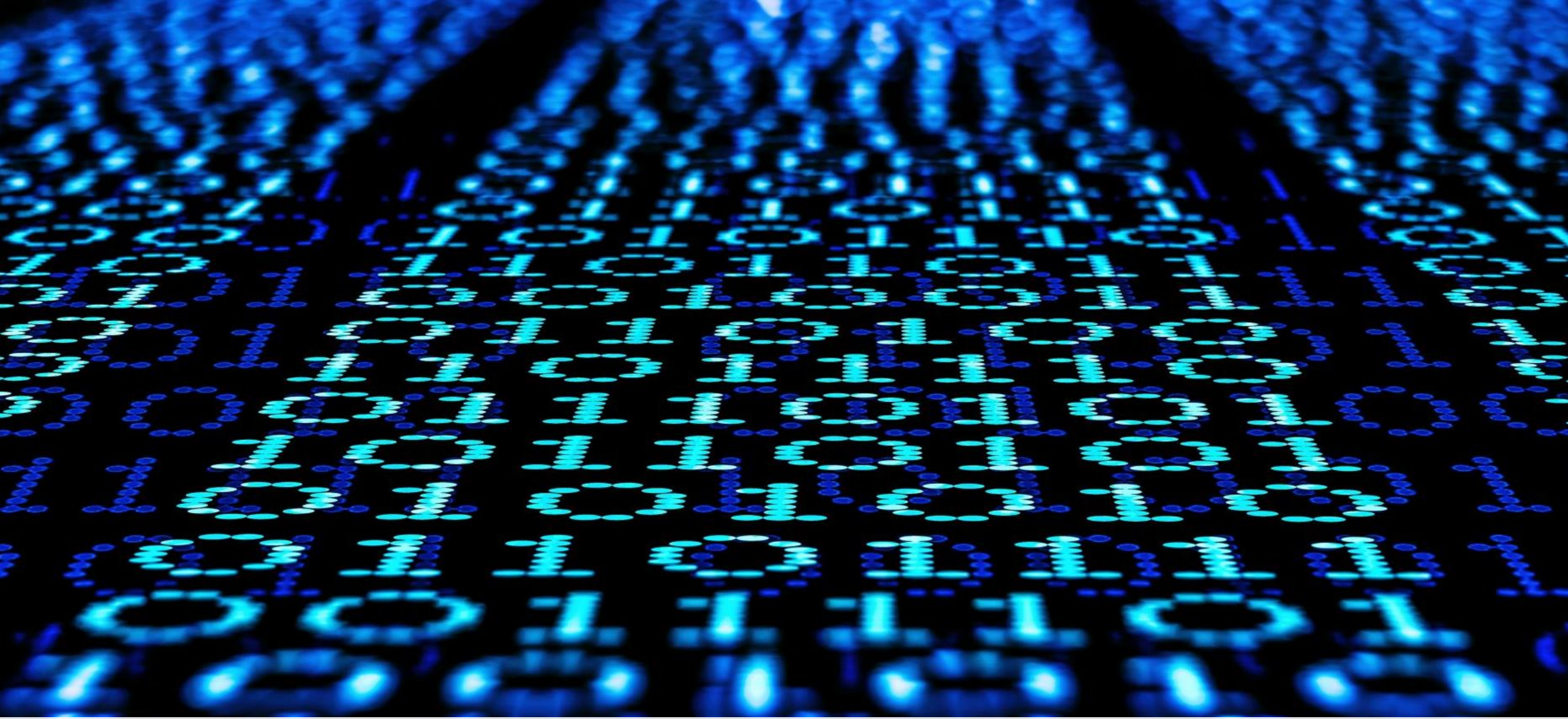

alarm_off
alarm_off()
high_pressure_detection()
alarm_on
alarm_on()
setTimer(alarm_timer,60)
waiting
expire(alarm_timer)
reset(alarm_timer)

Block: alarm_actuator_driver


init
waiting
alarm_off()
alarm_on()
alarm_off
alarm_on



Take care in code I didn’t implement feature "storing pressure value in BlackBox"
Also I made alarm actuator and alarm monitor in one file



Code Implementation

datatype.h

```
/*
 *   Topic   :                                          Pressure Alarm Tracer
 *   File    :                                          __DATA_TYPE_H__
 *   Author  :                                          abdelfattahzakariaelbadry@gmail.com
 */

/*
 *   Custom Built In Data Types Definiation For Global Using
 */

// Header Protection
#ifndef __DATA_TYPE_H__
#define __DATA_TYPE_H__

typedef volatile unsigned char                         vuint8_t;
typedef volatile signed char                           vsint8_t;
typedef volatile unsigned short int                    vuint16_t;
typedef volatile signed short int                      vsint16_t;
typedef volatile unsigned long int                     vuint32_t;
typedef volatile signed long int                       vsint32_t;
typedef volatile unsigned long long int                vuint64_t;
typedef volatile signed long long int                  vsint64_t;
typedef volatile float                                 vfloat32_t;
typedef volatile double                                vdouble64_t;
typedef volatile long double                           vdouble96_t;

#define Element_Type                                   vuint32_t
#define True                                           1
#define False                                          0

#endif
```

driver.h

```c
/*
 *   Topic   :                                  Pressure Alarm Tracer
 *   File    :                                  __Driver_H__
 *   Author  :                                  abdelfattahzakariaelbadry@gmail.com
*/

// Header Protection
#ifndef __Driver_H__
#define __Driver_H__
#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 ();


#endif
```

driver.c

```c
/*
 *   Topic    :                                    Pressure Alarm Tracer
 *   File     :                                    __Driver_C__
 *   Author   :                                    abdelfattahzakariaelbadry@gmail.com
 */

#include "driver.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;
}
```

state.h

```c
/*
 *   Topic   :                                                  Pressure Alarm Tracer
 *   File    :                                                  __STATE_H__
 *   Author  :                                                  abdelfattahzakariaelbadry@gmail.com
*/


/* * This Header File Aims To Define Smart Prototype Functions Macro For Differrent Useable Functions Names*/

// Header Protection
#ifndef __STATE_H__
#define __STATE_H__


#include "datatype.h"

/* *Smart Prototype Functions Macros*/
#define ST_define(_stateFunc_)                                  void ST_##_stateFunc_()
// Macro To Generate | Get Only The Function To Assigin It To A Pointer To Func
#define State(_stateFunc_)                                      ST_##_stateFunc_
// Macro To Call A Function
#define State_Call(_stateFunc_)                                  ST_##_stateFunc_()


/* *Driver.h Useable API Interfaces*/
void Delay(int nCount);
int getPressureVal();
void Set_Alarm_actuator(int i);
void GPIO_INITIALIZATION ();


#endif
```

pressure.h

```c
/*
 *   Topic   :                                                    Pressure Alarm Tracer
 *   File    :                                                    __PRESSURE_H__
 *   Author  :                                                    abdelfattahzakariaelbadry@gmail.com
*/

/* * This Header File Aims To Decleare The Pressure Sensors Tracer Prototype Functions And There Associated Structure*/

// Header Protection
#ifndef __PRESSURE_H__
#define __PRESSURE_H__
#include "state.h"

enum
{
    WAITING ,
    READING ,
    INIT
}ST_Preasure_Sensor_State;


// Decleare A Pointer To A Pressure Sensor Function
void (*Ptr_Pressure_Sensor_Func) (void);


// Define The Essential Function Associated With Pressure Sensor Jop Using A Smart Macro
ST_define(INIT);
// void ST_2()

ST_define(WAITING);
// void ST_0()

ST_define(READING);
// void ST_1()
#endif
```

pressure.c

```c
/*
 *  Topic   :                                                   Pressure Alarm Tracer
 *  File    :                                                   __PRESSURE_C__
 *  Author  :                                                   abdelfattahzakariaelbadry@gmail.com
*/


/* * This Header File Aims To Define The Pressure Sensors Tracer Prototype Functions Body*/


#include "pressure_sensor.h"


// Define The Global Useable Variables Which Will Later Be Externed Into Main.c
vuint32_t threshold= 20;
vuint32_t pressure_value= 0;


// Define A Pointer To A Pressure Sensor Function
void (*Ptr_Pressure_Sensor_Func) (void);

// Define The Prototypes Functions Body
ST_define(INIT)
{
    // Sate Id | Name: INIT-->> 2
    ST_Preasure_Sensor_State= INIT;

    //Sate Action: Assign Ptr_Pressure_Sensor_Func With The Address Of Initial | Startup | Power On Sate Function: ST_define(INIT) -->> ST_2
    Ptr_Pressure_Sensor_Func= State(INIT);

    return;
}
```

Pressure.c cont..

```
35
36  ST_define(WAITING)
37  {
38      vsint32_t i;
39
40      // Sate Id | Name: -->> 0
41      ST_Preasure_Sensor_State= WAITING;
42
43      // Assign Ptr_Pressure_Sensor_Func With Current State Function: ST_define(WAITING) -->> ST_0
44      Ptr_Pressure_Sensor_Func= State(WAITING);
45
46      // Sate Action: Making A Delay Using Custom Tradtional Empty Looping
47      for(i= 5000; i != 0; i--);
48
49      return;
50  }
51
52
53  ST_define(READING)
54  {
55      // Sate Id | Name: INIT-->> 1
56      ST_Preasure_Sensor_State= READING;
57
58      //Sate Action: Get Sensor Reads
59      // Read Using get_pressure_val API
60      pressure_value= getPressureVal();
61
62      // Making A Delay
63      State_Call(WAITING);
64
65      // Assign Ptr_Pressure_Sensor_Func With Current State Function: ST_define(WAITING) -->> ST_0
66      Ptr_Pressure_Sensor_Func= State(READING);
67
68      return;
69  }
```

alarm.h

```c
/*
 *   Topic   :                                                  Pressure Alarm Tracer
 *   File    :                                                  __ALARM_H__
 *   Author  :                                                  abdelfattahzakariaelbadry@gmail.com
*/


/* * This Header File Aims To Decleare The Alarm Actuator Prototype Functions And There Associated Structure*/


// Header Protection
#ifndef __ALARM_H__
#define __ALARM_H__

#include "state.h"
enum
{
    ALARM_OFF ,
    ALARM_ON ,
    ALARM_INIT
}ST_Alarm_Actuator_State;

// Decleare A Pointer To A Alarm Actuator Function
void (*Ptr_Alarm_Actuator_Func) (void);

// Define The Essential Function Associated With Alarm Actuator Jop Using A Smart Macro
ST_define(ALARM_INIT);
// void ST_2()

ST_define(ALARM_OFF);
// void ST_0()

ST_define(ALARM_ON);
// void ST_1()
#endif
```

alarm.c

```c
/*
 *  Topic   :                                   Pressure Alarm Tracer
 *  File    :                                   __ALARM_C__
 *  Author  :                                   abdelfattahzakariaelbadry@gmail.com
*/


/* * This Header File Aims To Define The Alarm Actuator Prototype FunctiALARM_ONs Body*/


#include "alarm.h"


// Define The Local File Static Variables
static vsint32_t count= 2500;


// Define A Pointer To A Alarm Actuator FunctiALARM_ON
void (*Ptr_Alarm_Actuator_Func) (void);


// Define The Prototypes FunctiALARM_ONs Body
ST_define(ALARM_INIT)
{
    // Sate Id | Name: ALARM_INIT-->> 2
    ST_Alarm_Actuator_State= ALARM_INIT;

    //Sate ActiALARM_ON: Assign Ptr_Alarm_Actuator_Func With The Address Of ALARM_INITial | Startup | Power ALARM_ON Sate FunctiALARM_ON: ST_define(ALARM_INIT)
    Ptr_Alarm_Actuator_Func= State(ALARM_INIT);

    return;
}

```

alarm.c cont..

```
34  ST_define(ALARM_OFF)
35  {
36      // Sate Id | Name: ALARM_OFF-->> 0
37      ST_Alarm_Actuator_State= ALARM_OFF;
38
39      // State ActiALARM_ON: Diseable Alarm Actuator
40      // Using set_alarm_actuator API
41      Set_Alarm_actuator(1);
42
43      // Assign Ptr_Alarm_Actuator_Func With Current State FunctiALARM_ON: ST_define(ALARM_OFF) -->> ST_0
44      Ptr_Alarm_Actuator_Func= State(ALARM_OFF);
45
46      return;
47  }
48  ST_define(ALARM_ON)
49  {
50      // Sate Id | Name: ALARM_ON-->> 1
51      ST_Alarm_Actuator_State= ALARM_ON;
52
53      // State ActiALARM_ON: enable Alarm Actuator
54      // Using set_alarm_actuator API
55      Set_Alarm_actuator(0);
56
57      // Making A Delay Using Delay Api
58      Delay(count);
59
60      // Turn ALARM_OFF Alarm Actuator Again After The Delay Period
61      // Using set_alarm_actuator API
62      Set_Alarm_actuator(1);
63
64      // Assign Ptr_Alarm_Actuator_Func With Current State FunctiALARM_ON: ST_define(ALARM_OFF) -->> ST_0
65      Ptr_Alarm_Actuator_Func= State(ALARM_OFF);
66      return;
67  }
```

main.c

```c
/*
 *   Topic   :                                                  Pressure Alarm Tracer
 *   File    :                                                  __MAIN_C__
 *   Author  :                                                  abdelfattahzakariaelbadry@gmail.com
*/

// Including Essential Sensors | Actuators | MC Header Files
#include <stdint.h>
#include "driver.h"
#include "pressure_sensor.h"
#include "alarm.h"


// Include Common Operational Variables Aomng Differrent Components
extern vuint32_t threshold;
extern vuint32_t pressure_value;

// Define The Setup Function To Init Differrent Components Into Init | Startup | Power On State
void setup(void)
{
    // Enable | Init GPIO Target Port
    GPIO_INITIALIZATION();

    // Init Pressure Sensor Into Init | Startup | Power On State
    Ptr_Pressure_Sensor_Func= State(INIT);
    Ptr_Pressure_Sensor_Func();

    // Init Alarm Actuator Into Init | Startup | Power On State
    Ptr_Alarm_Actuator_Func= State(ALARM_INIT);
    Ptr_Alarm_Actuator_Func();

    return;
}

```

main.c cont..

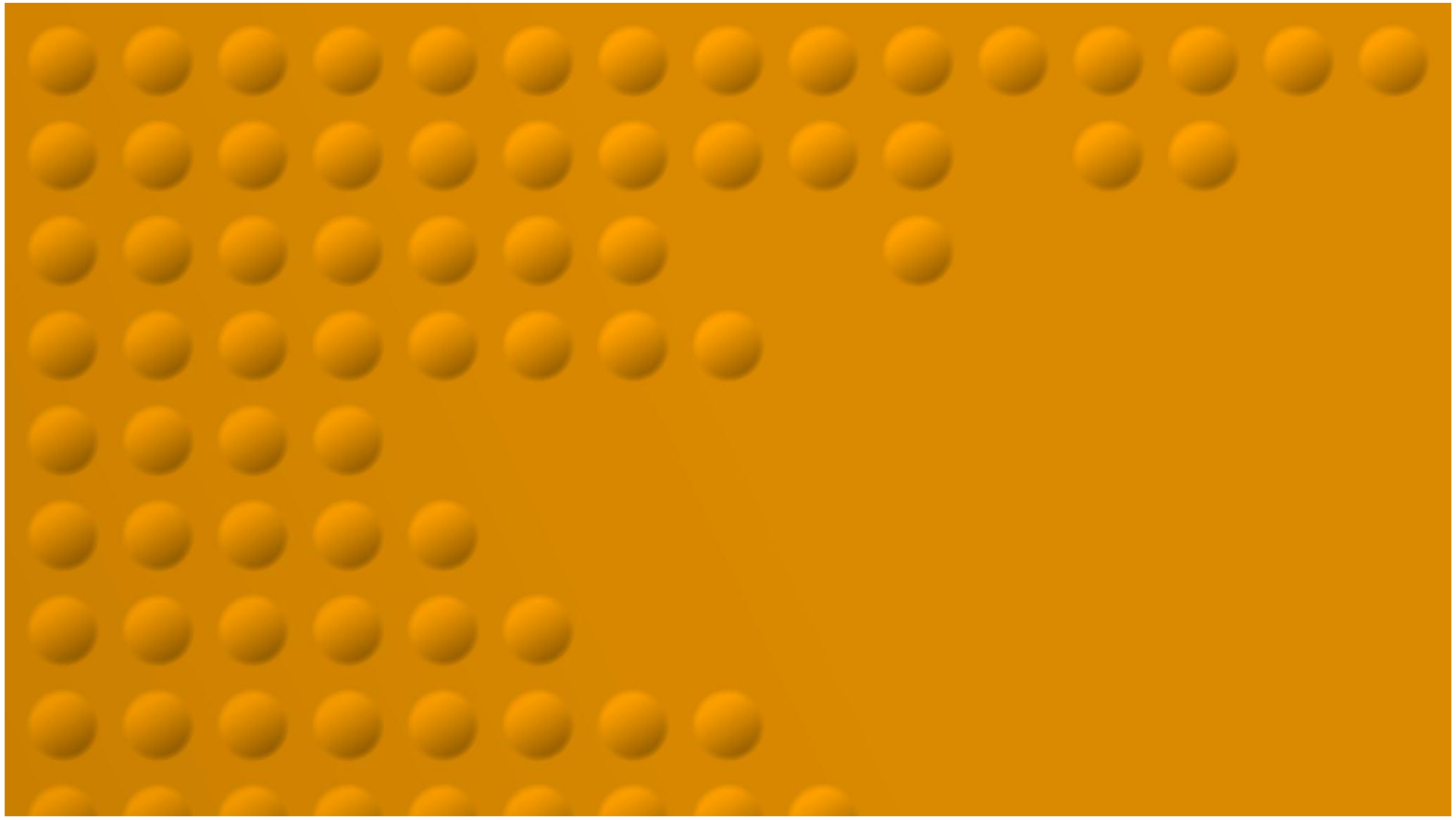
```c
int main(void)
{
    setup();

    while(True)
    {
        // Get Pressure Sensor Reads
        Ptr_Pressure_Sensor_Func();

        // Check Up Reads Aganist Tareget Threshold
        if(pressure_value <= threshold)
        {
            // Enable Alarm Actuator For A Time Period Then Turn OFF It
            // a: Assign Ptr_Alarm_Actuator_Func With Alarm On Function Address
            Ptr_Alarm_Actuator_Func= State(ALARM_ON);
            // b: Jump To Function Block Ins
            Ptr_Alarm_Actuator_Func();
        }
    }

    return 0;
}
```

Pass Through Excutable File

Excutable File Sections

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .TEXT         00000400  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, CODE
  1 .DATA         00000008  20000000  08000400  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .BSS          000003f0  20000008  08000408  00010008  2**2
                  ALLOC
  3 .debug_info   00000646  00000000  00000000  00010008  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 0000037b  00000000  00000000  0001064e  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000298  00000000  00000000  000109c9  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000a0  00000000  00000000  00010c61  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   000002a2  00000000  00000000  00010d01  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    00000293  00000000  00000000  00010fa3  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  00011236  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00011247  2**0
                  CONTENTS, READONLY
 11 .debug_frame  000001cc  00000000  00000000  0001127c  2**2
                  CONTENTS, READONLY, DEBUGGING
 12 .bss          00000010  200003f8  200003f8  000103f8  2**2
                  ALLOC
```

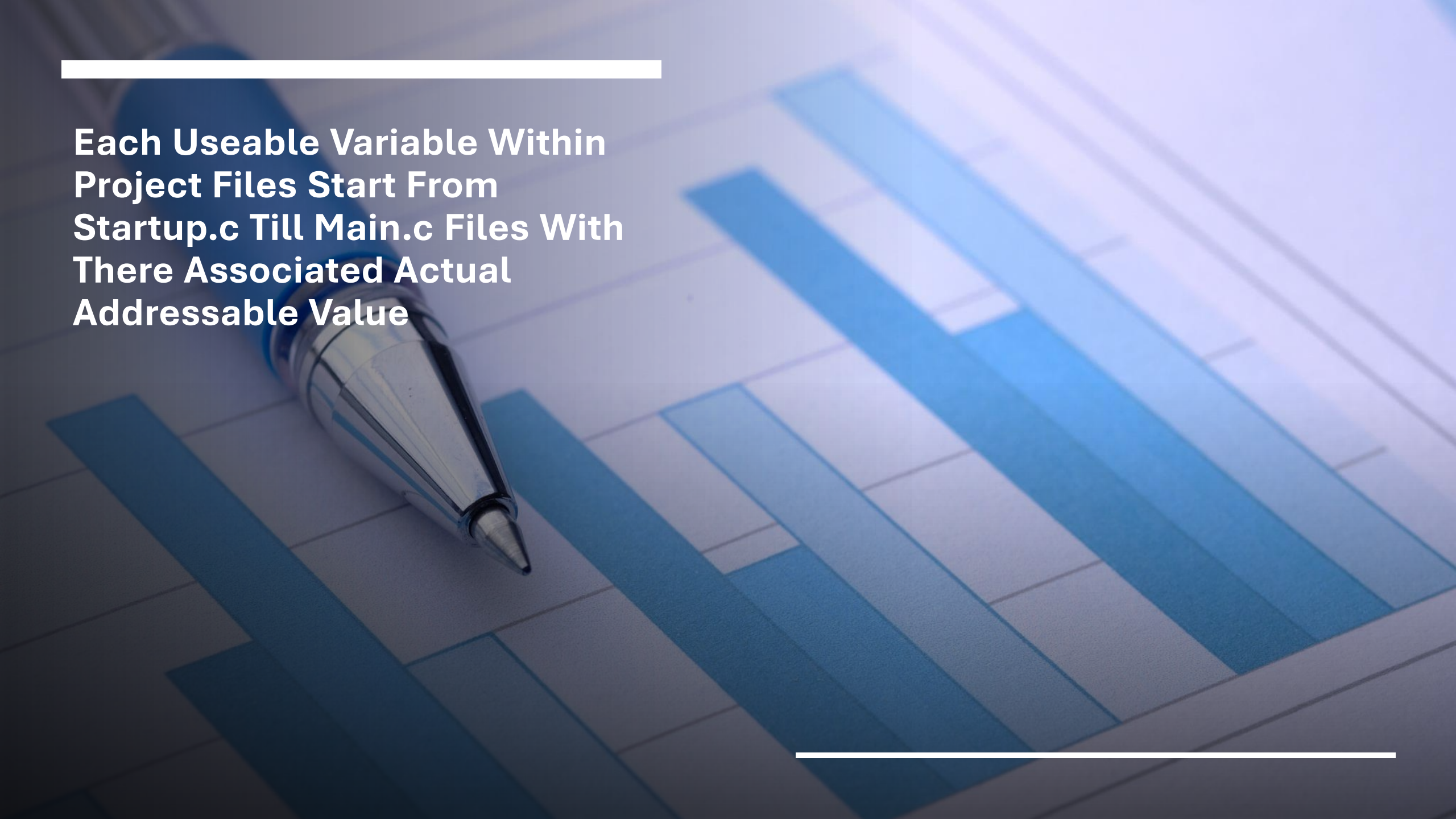
Some Of Extuable Dot Text Section Essambly Instructions



```
Pressure_Tracer.elf:     file format elf32-littlearm


Disassembly of section .TEXT:

08000000 <vectors>:
 8000000:       f8 03 00 20 19 03 00 08 f5 03 00 08 f5 03 00 08     ... ............
 8000010:       f5 03 00 08 f5 03 00 08 f5 03 00 08                 ............

0800001c <ST_ALARM_INIT>:
 800001c:       b480            push    {r7}
 800001e:       af00            add     r7, sp, #0
 8000020:       f240 33fc       movw    r3, #1020       ; 0x3fc
 8000024:       f2c2 0300       movt    r3, #8192       ; 0x2000
 8000028:       f04f 0202       mov.w   r2, #2
 800002c:       701a            strb    r2, [r3, #0]
 800002e:       f240 33f8       movw    r3, #1016       ; 0x3f8
 8000032:       f2c2 0300       movt    r3, #8192       ; 0x2000
 8000036:       f240 021d       movw    r2, #29
 800003a:       f6c0 0200       movt    r2, #2048       ; 0x800
 800003e:       601a            str     r2, [r3, #0]
 8000040:       bf00            nop
 8000042:       46bd            mov     sp, r7
 8000044:       bc80            pop     {r7}
 8000046:       4770            bx      lr
```

Some Information Associated With Target MC | SOC | Porda Such As OS , Where Dot Text Is Allocated ,...

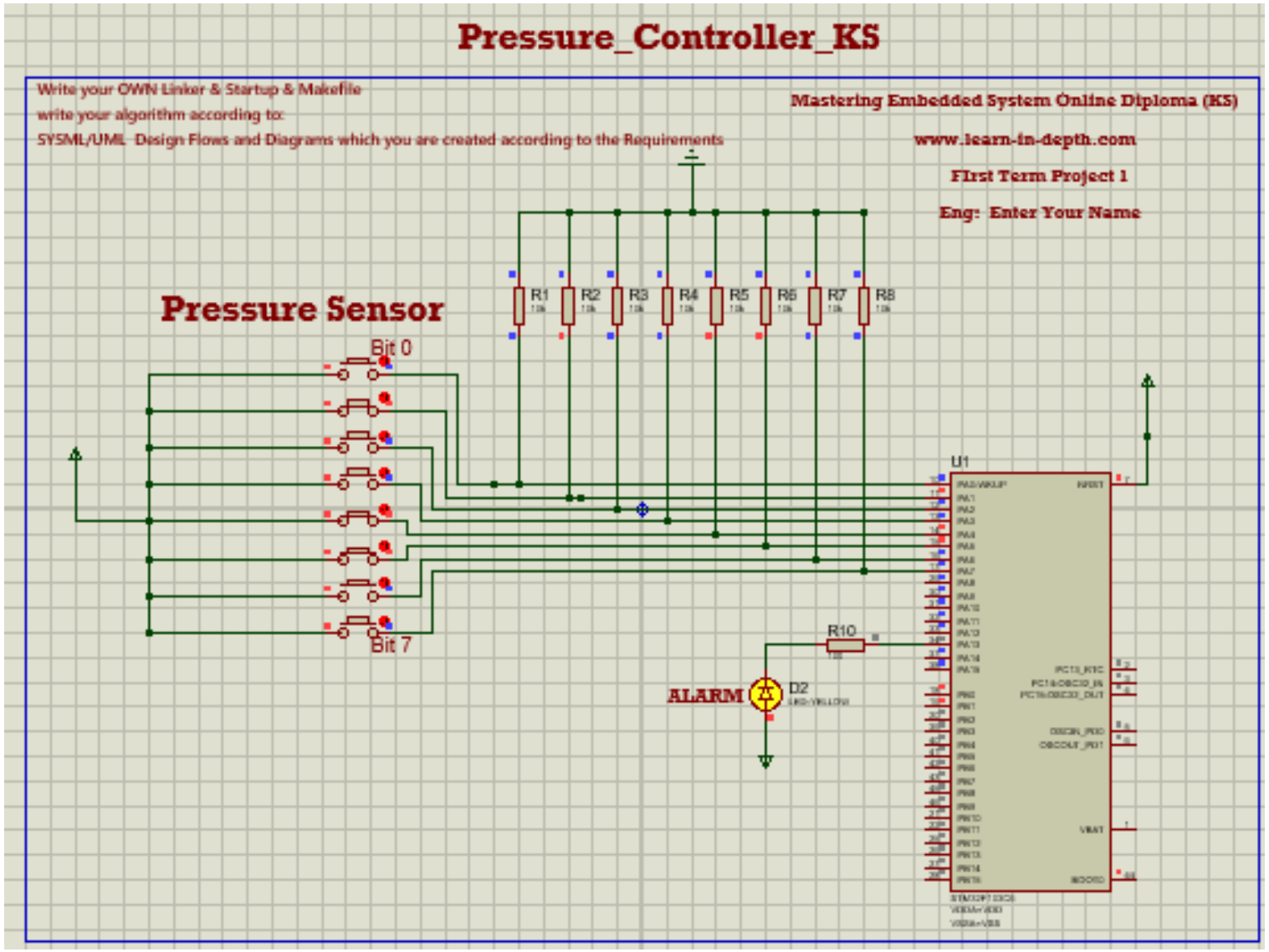
```
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          52 (bytes into file)
  Start of section headers:          70892 (bytes into file)
  Flags:                             0x5000000, Version5 EABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         3
  Size of section headers:           40 (bytes)
  Number of section headers:         17
  Section header string table index: 14
```



Each Useable Variable Within Project Files Start From Startup.c Till Main.c Files With There Associated Actual Addressable Value

```
$ arm-none-eabi-nm.exe Pressure_Tracer.elf
080003f4 W BUS_FAULT
20000000 d count
080003f4 T DEFAULT_HANDLER
080000c0 T Delay
20000010 B E_BSS
20000008 D E_DATA
08000400 T E_TEXT
080000e4 T getPressureVal
0800014c T GPIO_INITIALIZATION
080003f4 W HARD_FAULT
2000000c B i
08000214 T main
080003f4 W MEM_MANAGE
080003f4 W NMI
20000008 B pressure_value
20003f8 B Ptr_Alarm_Actuator_Func
20000404 B Ptr_Pressure_Sensor_Func
08000318 T RESET
20000008 B S_BSS
20000000 D S_DATA
080000fc T Set_Alarm_actuator
080001cc T setup
20003fc B ST_Alarm_Actuator_State
0800001c T ST_ALARM_INIT
08000048 T ST_ALARM_OFF
08000078 T ST_ALARM_ON
08000264 T ST_INIT
20000400 B ST_Preasure_Sensor_State
080002d8 T ST_READING
08000290 T ST_WAITING
20003f8 B STACK_TOP
20000004 D threshold
080003f4 W USGE_FAULT
08000000 T vectors
```

Simulation Results



Pressure More Than Or Equal Threshold

Author:
abdelfattahzakariaelbadry@gmail.com