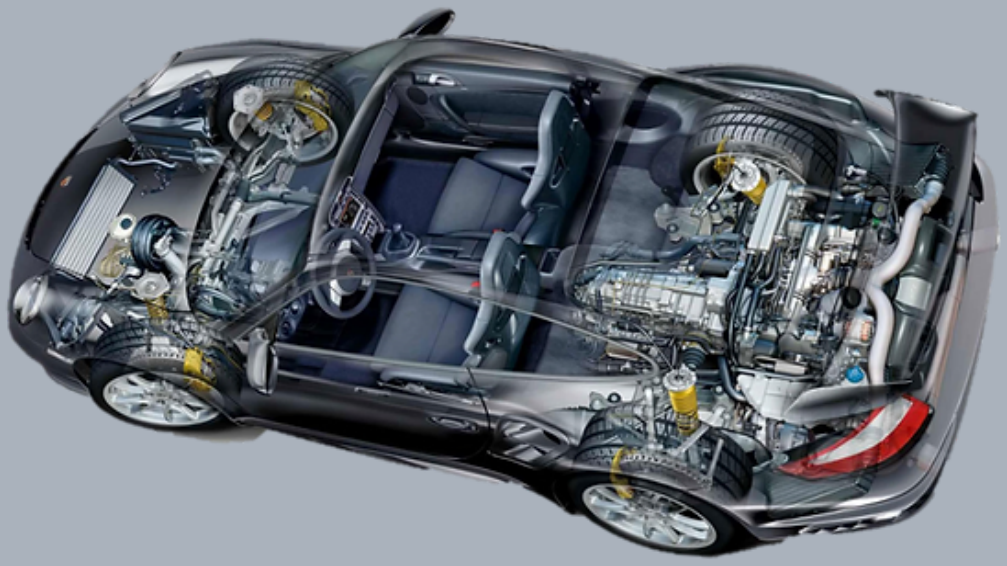


MASTERING EMBEDDED SYSTEM ONLINE DIPLOMA
WWW.LEARN-IN-DEPTH.COM

FIRST TERM { FINAL PROJECT 1 }

[HIGH PRESSURE DETECTION SYSTEM]

Eng.

MOHAMED NABIL MOHAMED

My Profile:

@LEARN-IN-DEPTH.COM/M70.ENG

System Architecting Steps

01 Case Study

02 Method

03 Partitioning

04 System Requirements

05 System Analysis

06 System Design

Case Study

Specifications (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.
- keeps track of the measured values.

Assumptions:

- The controller set up and shutdown procedures are not modeled.
- The controller maintenance is not modeled.
- The pressure sensor never fails.
- The alarm never fails.
- The controller never faces power cut.

Versioning :

The “keep track of measured value” option is not modeled in the first version of the design

Method

Waterfall Method

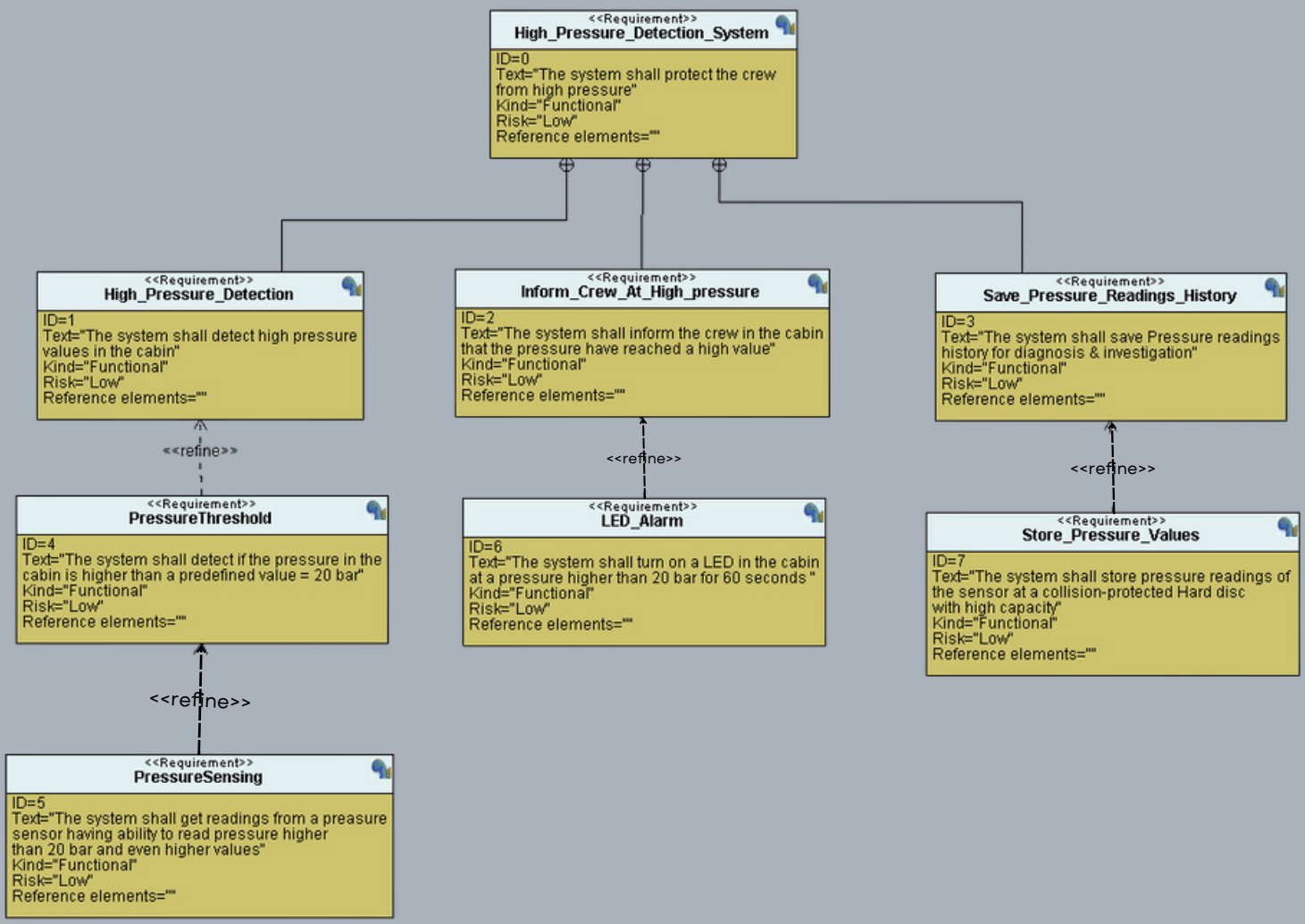
The Waterfall method is the best choice for our High Pressure Detection System due to its structured and sequential approach. It ensures meticulous planning, comprehensive documentation, effective risk management, and thorough testing. These features are crucial for developing a system that requires accuracy, safety, and reliability in high-pressure environments Such as the plane navigation cabin.

Partitioning

In the System it was determined that the ship that will be used to control the High pressure detection operation is : STM32F103C6

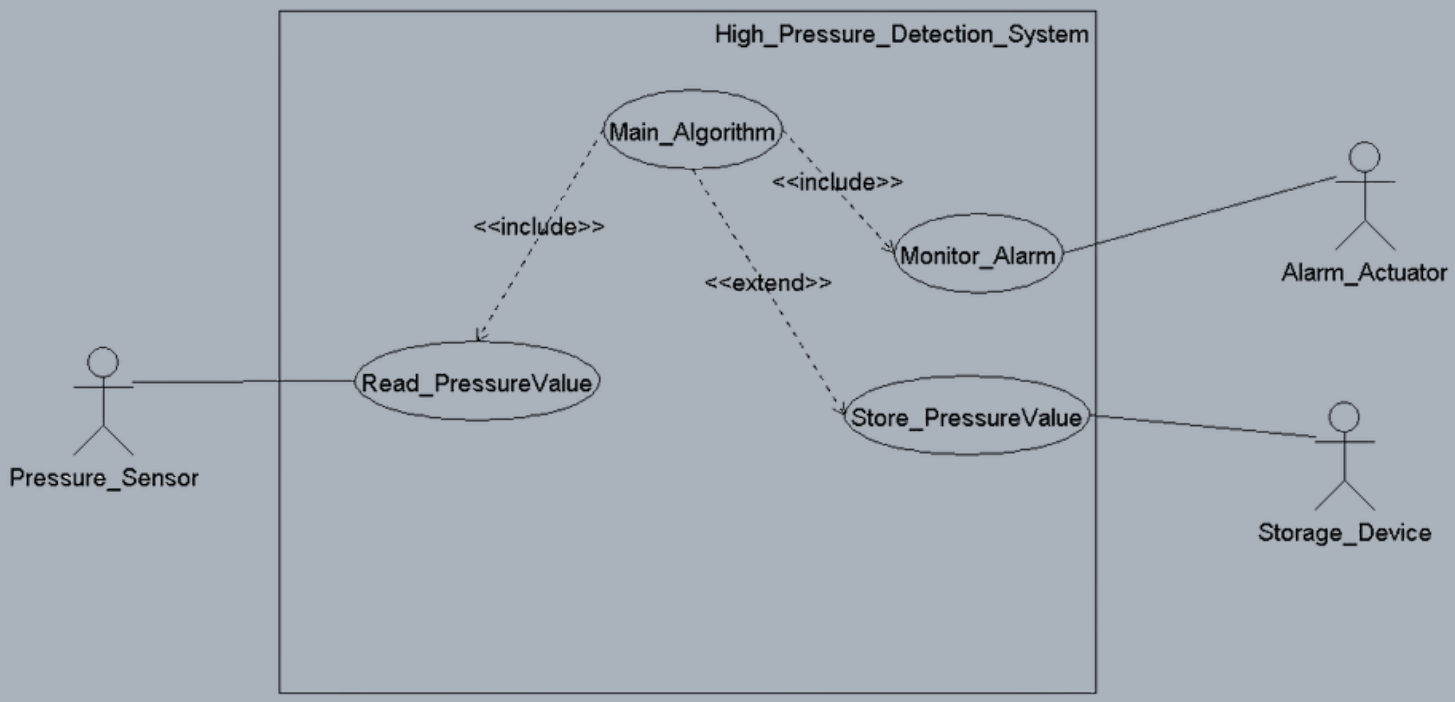
System Requirements

Requirement Diagram:

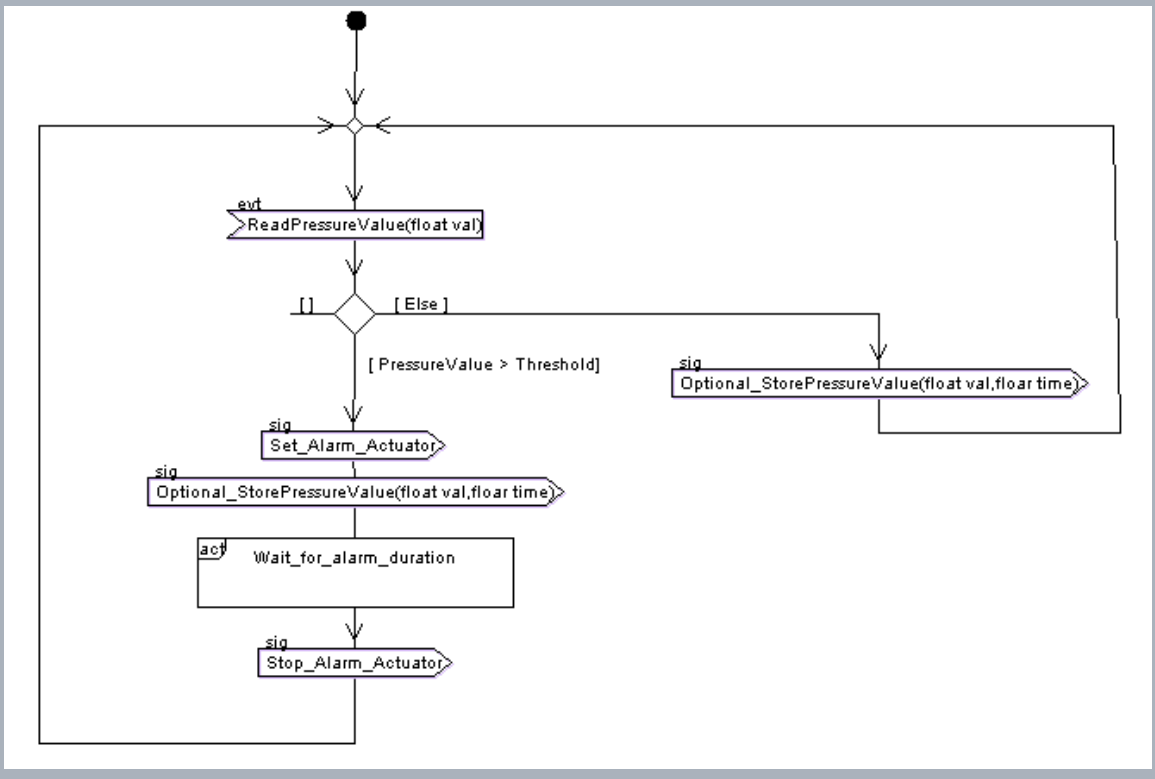


System Analysis

Use Case Diagram

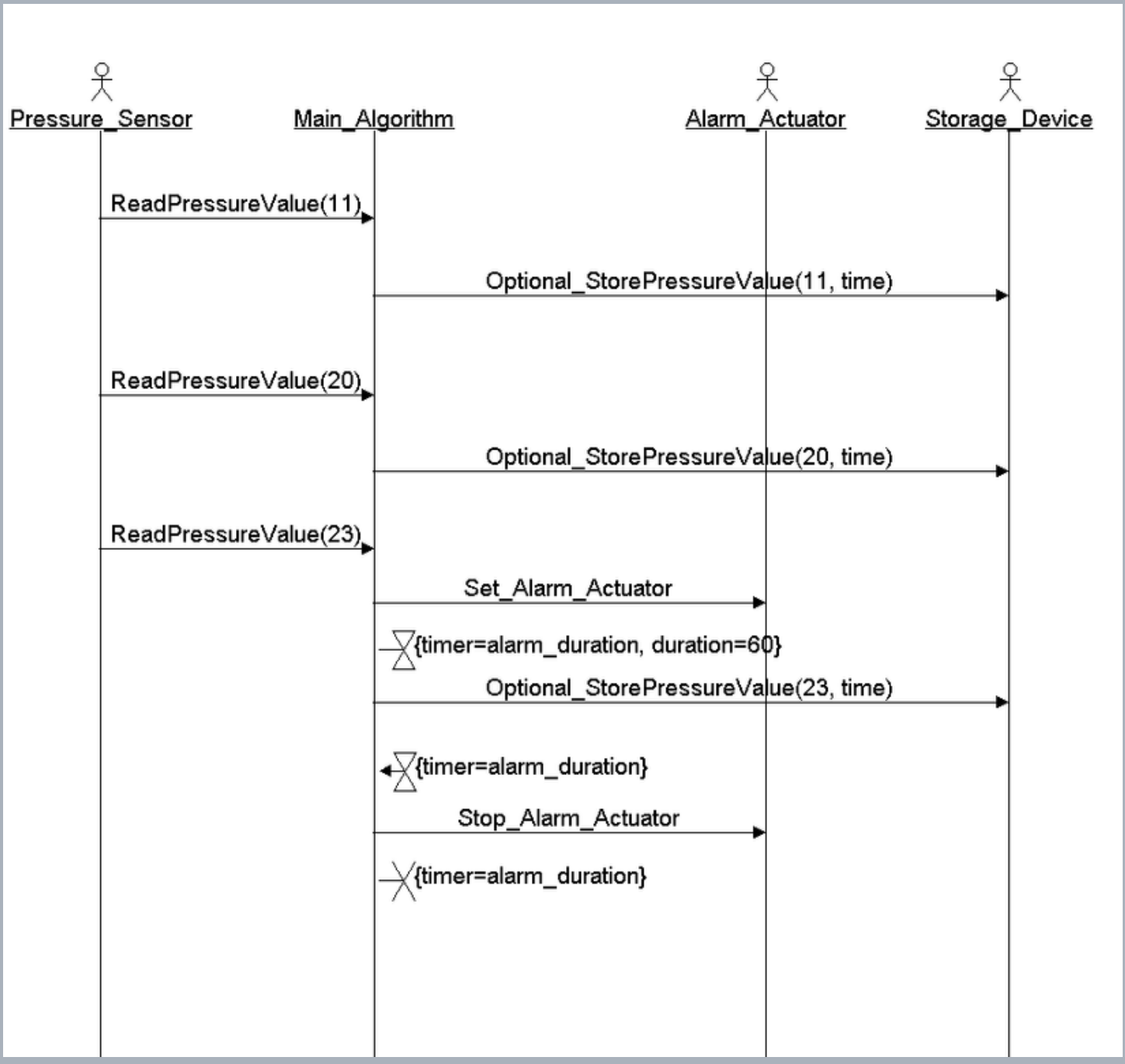


Activity Diagram



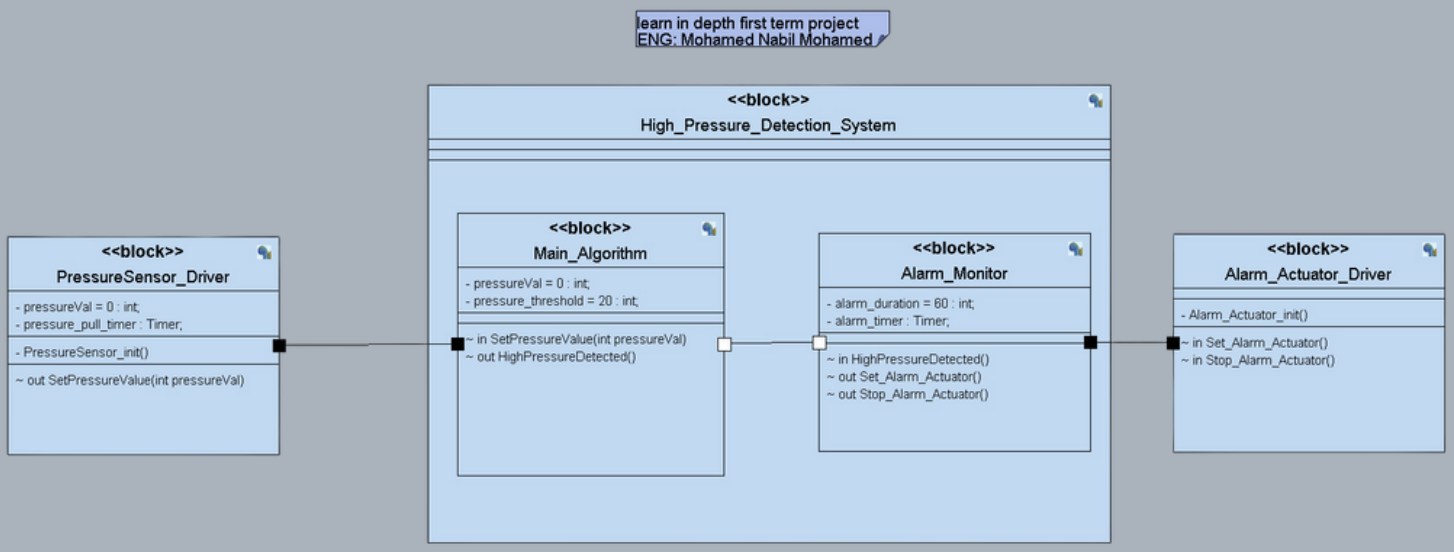
System Analysis

Sequence Diagram



System Design

System Block Diagram



State Machine Diagrams:

1. Pressure Sensor Driver.
2. Main Algorithm.
3. Alarm Monitor.
4. Alarm Actuator Driver.

System Design

Image for each file.C & file.h with the Corresponding state machine:

1-Pressure Sensor Driver.

Psensor_Driver.c

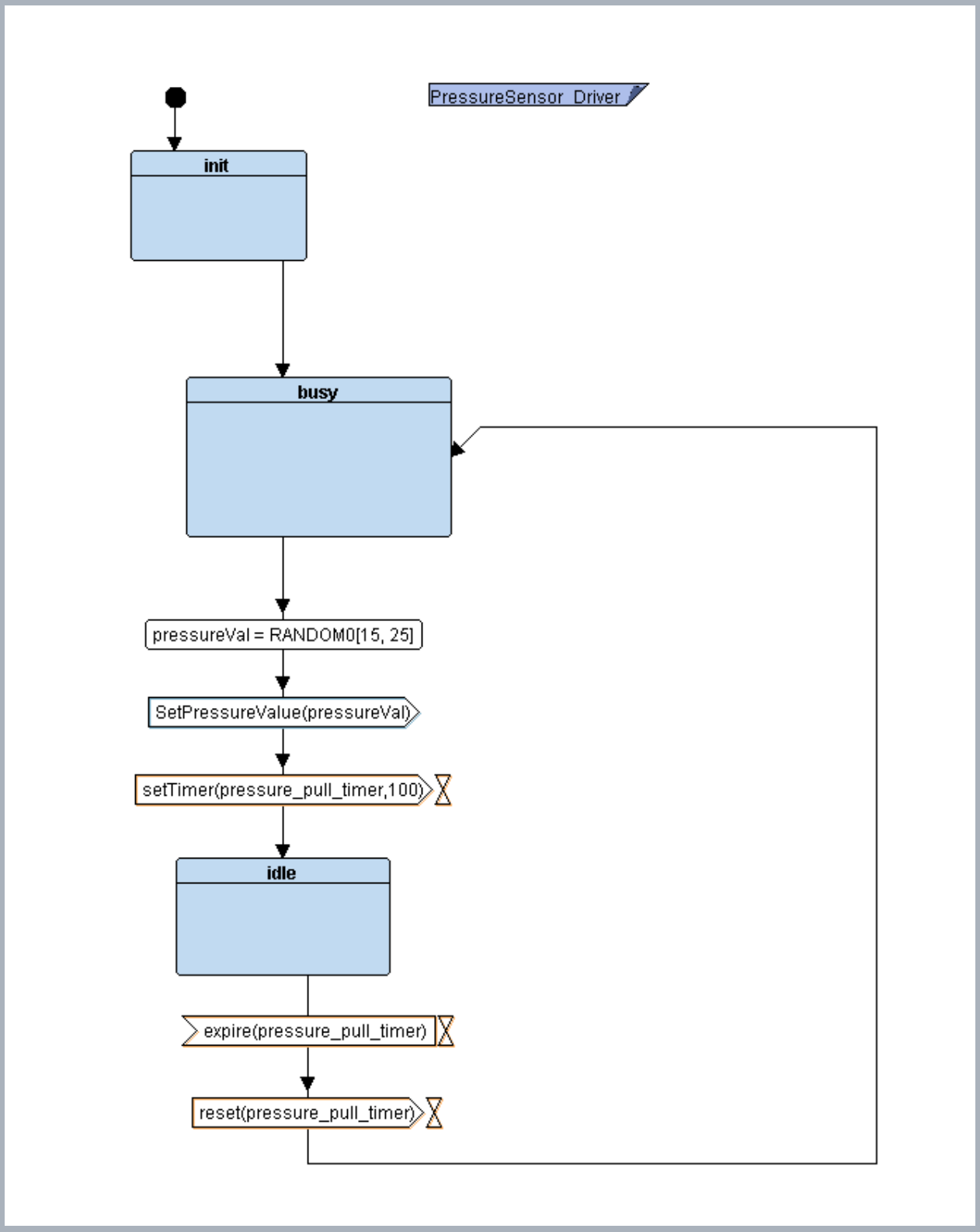
```c
//Includes
#include "Psensor_Driver.h"
#include "driver.h"

//Module Variables
int Psensor_pressureVal = 0;

extern void (*PSensor_state)();

//Functions

void PressureSensor_init(){

    //Set first state
    PSensor_state = STATE(PSensor_busy);

}

STATE_DEFINE(PSensor_busy){

    //Read pressure value from pressure sensor
    Psensor_pressureVal = getPressureVal();

    //Send signal (PressureSensor_Driver ---> Main_Algorit
    SetPressureValue(Psensor_pressureVal);

    //Update state
    PSensor_state = STATE(PSensor_idle);

}

STATE_DEFINE(PSensor_idle){

    //Delay between readings
    Delay(Psensor_pressure_pull_time);

    //Set first state
    PSensor_state = STATE(PSensor_busy);

}
```

Psensor_Driver.h

```c
/*
 * Psensor_Driver.h
 *
 *  Created on: Aug 29, 2023
 *      Code By: Mohamed Nabil Mohamed
 */

#ifndef PSENSOR_DRIVER_H_
#define PSENSOR_DRIVER_H_

//Defines
#define Psensor_pressure_pull_time 6000

//Includes
#include "state.h"

//State IDs
enum{
    PSensor_busy,
    PSensor_idle
}PSensor_state_id;


//Prototypes
void PressureSensor_init();
STATE_DEFINE(PSensor_busy);
STATE_DEFINE(PSensor_idle);


//State pointer
void (*PSensor_state)();


#endif /* PSENSOR_DRIVER_H_ */
```

System Design

Pressure Sensor Driver State Machine



System Design

2-Main Algorithm.

Main_Algorithm.c

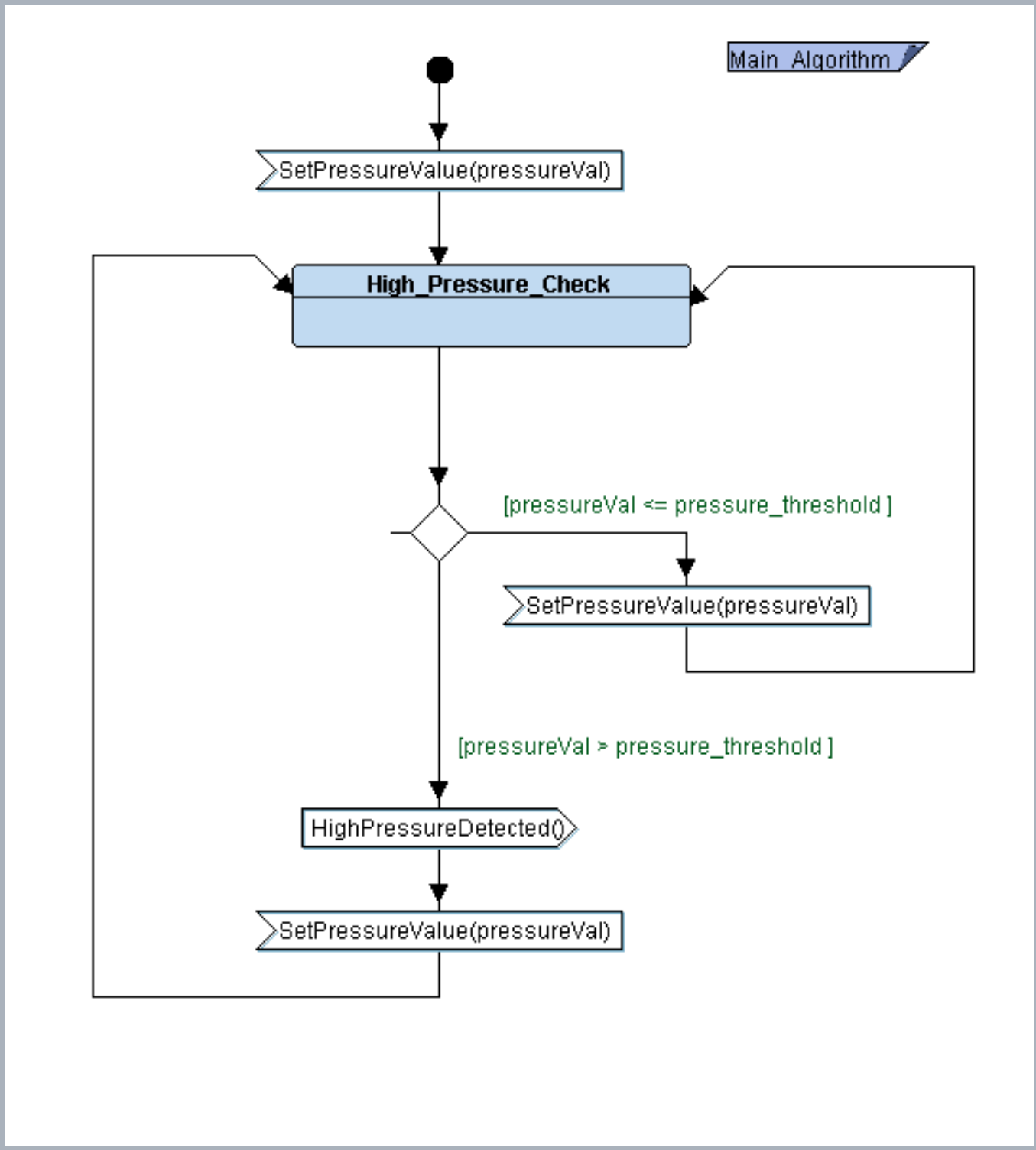
```c
//Includes
#include "Main_Algorithm.h"

//Module Variables
int Main_pressureVal = 0;
int Main_pressure_threshold = Pthreshold;

extern void (*Main_Algorithm_state)();

//Functions


/*PressureSensor_Driver ---> Main_Algorithm*/
void SetPressureValue(int pressureVal){

    Main_pressureVal = pressureVal;
    Main_Algorithm_state = STATE(High_Pressure_Check);

}

STATE_DEFINE(High_Pressure_Check){

    //Receive pressure reading signal (PressureSensor_

    //State ID
    Main_Algorithm_state_id = High_Pressure_Check;


    //Main Algorithm
    if(Main_pressureVal > Main_pressure_threshold ){

        //Main_Algorithm ---> Alarm_Monitor
        HighPressureDetected();

    }

    //Update state
    Main_Algorithm_state = STATE(High_Pressure_Check);

}
```

Main_Algorithm.h

```c
/*
 * Main_Algorithm.h
 *
 *  Created on: Aug 29, 2023
 *      Code By: Mohamed Nabil Mohamed
 */

#ifndef MAIN_ALGORITHM_H_
#define MAIN_ALGORITHM_H_

//Includes
#include "state.h"

//Defines

//select Pressure threshold value in bars
#define Pthreshold 20

//State IDs
enum{
    High_Pressure_Check
}Main_Algorithm_state_id;


//Prototypes
STATE_DEFINE(High_Pressure_Check);


//State pointer
void (*Main_Algorithm_state)();


#endif /* MAIN_ALGORITHM_H_ */
```

System Design

Main Algorithm State Machine



System Design

3-Alarm Monitor.

Alarm_Monitor.c

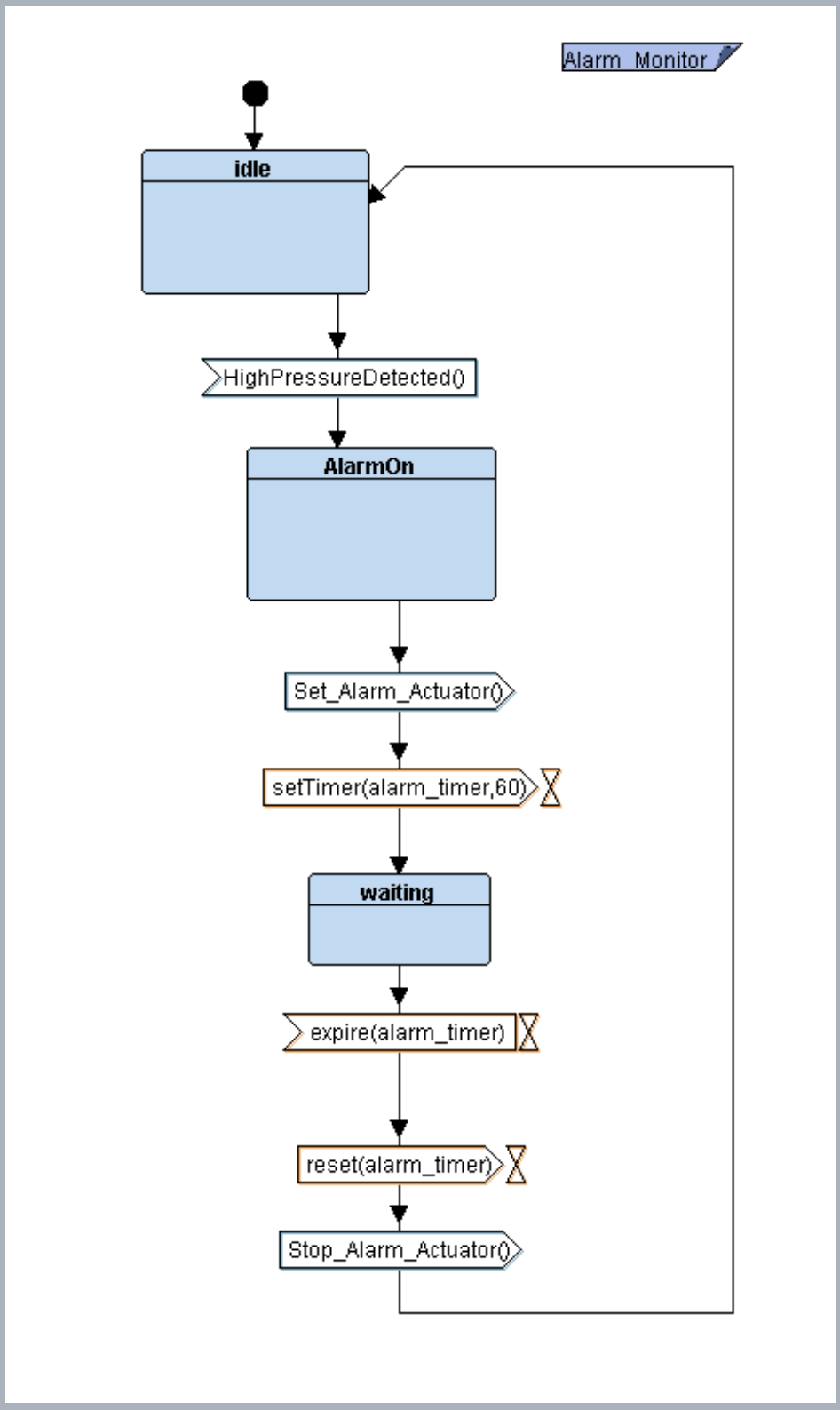
```c
//Includes
#include "Alarm_Monitor.h"
#include"driver.h"

//Module Variables
extern void (*Alarm_Monitor_state)();

//Functions

//Main_Algorithm ---> Alarm_Monitor
void HighPressureDetected(){
    Alarm_Monitor_state = STATE(Alarm_Monitor_AlarmOn);
}


STATE_DEFINE(Alarm_Monitor_AlarmOff){
    Alarm_Monitor_state_id = Alarm_Monitor_AlarmOff;
    //Do nothing until high pressure is detected
    }

STATE_DEFINE(Alarm_Monitor_AlarmOn){

    //State ID
    Alarm_Monitor_state_id = Alarm_Monitor_AlarmOn;

    //Alarm_Monitor ---> Alarm_Actuator_Driver
    Set_Alarm_Actuator();

    //set next state
    Alarm_Monitor_state = STATE(Alarm_Monitor_waiting);
}

STATE_DEFINE(Alarm_Monitor_waiting){

    Alarm_Monitor_state_id = Alarm_Monitor_waiting;

    //wait for alarm_duration
    Delay(alarm_duration);

    //stop alarm
    Stop_Alarm_Actuator();

    //set next state
    Alarm_Monitor_state = STATE(Alarm_Monitor_AlarmOff);
}
```

Alarm_Monitor.h

```c
#ifndef ALARM_MONITOR_H_
#define ALARM_MONITOR_H_

//Defines
#define alarm_duration 20000

//Includes
#include "state.h"

//States IDs
enum{
    Alarm_Monitor_AlarmOn,
    Alarm_Monitor_AlarmOff,
    Alarm_Monitor_waiting
}Alarm_Monitor_state_id;


//Prototypes
STATE_DEFINE(Alarm_Monitor_AlarmOn);
STATE_DEFINE(Alarm_Monitor_AlarmOff);
STATE_DEFINE(Alarm_Monitor_waiting);

//State pointer
void (*Alarm_Monitor_state)();



#endif /* ALARM_MONITOR_H_ */
```

System Design

Alarm Monitor State Machine



System Design

4-Alarm Actuator Driver Driver.

Alarm_Actuator_Driver.c

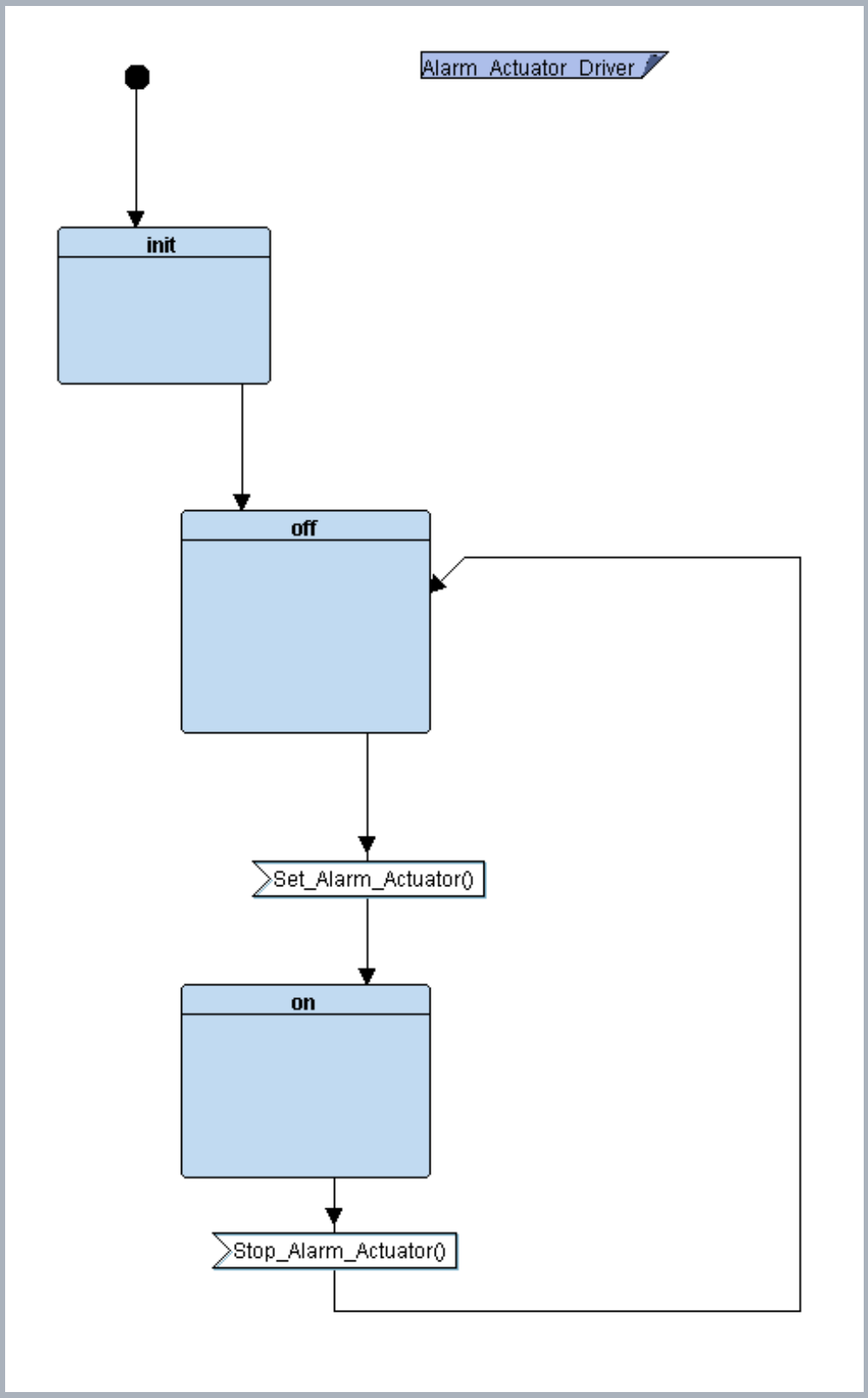
```c
//Includes
#include "Alarm_Actuator_Driver.h"
#include "driver.h"

//Module Variables
extern void (*Alarm_Actuator_state)();

//Functions

void Alarm_Actuator_init(){

    //Set first state
    Alarm_Actuator_state = STATE(Alarm_Actuator_off);
}

/* Alarm_Monitor ---> Alarm_Actuator_Driver */
void Set_Alarm_Actuator(){

    //Set equivalent state
    Alarm_Actuator_state = STATE(Alarm_Actuator_on);
}

void Stop_Alarm_Actuator(){
    //Set equivalent state
    Alarm_Actuator_state = STATE(Alarm_Actuator_off);
}


STATE_DEFINE(Alarm_Actuator_off){
    Set_Alarm_actuator(OFF);
}

STATE_DEFINE(Alarm_Actuator_on){
    Set_Alarm_actuator(ON);
}
```

Alarm_Actuator_Driver.h

```c
#ifndef ALARM_ACTUATOR_DRIVER_H_
#define ALARM_ACTUATOR_DRIVER_H_

//Defines
//according to given project connection (Acti
#define ON 0
#define OFF 1

//Includes
#include "state.h"

//State IDs
enum{
    Alarm_Actuator_off,
    Alarm_Actuator_on
}Alarm_Actuator_state_id;


//Prototypes
void Alarm_Actuator_init();
STATE_DEFINE(Alarm_Actuator_on);
STATE_DEFINE(Alarm_Actuator_off);


//State pointer
void (*Alarm_Actuator_state)();



#endif /* ALARM_ACTUATOR_DRIVER_H_ */
```

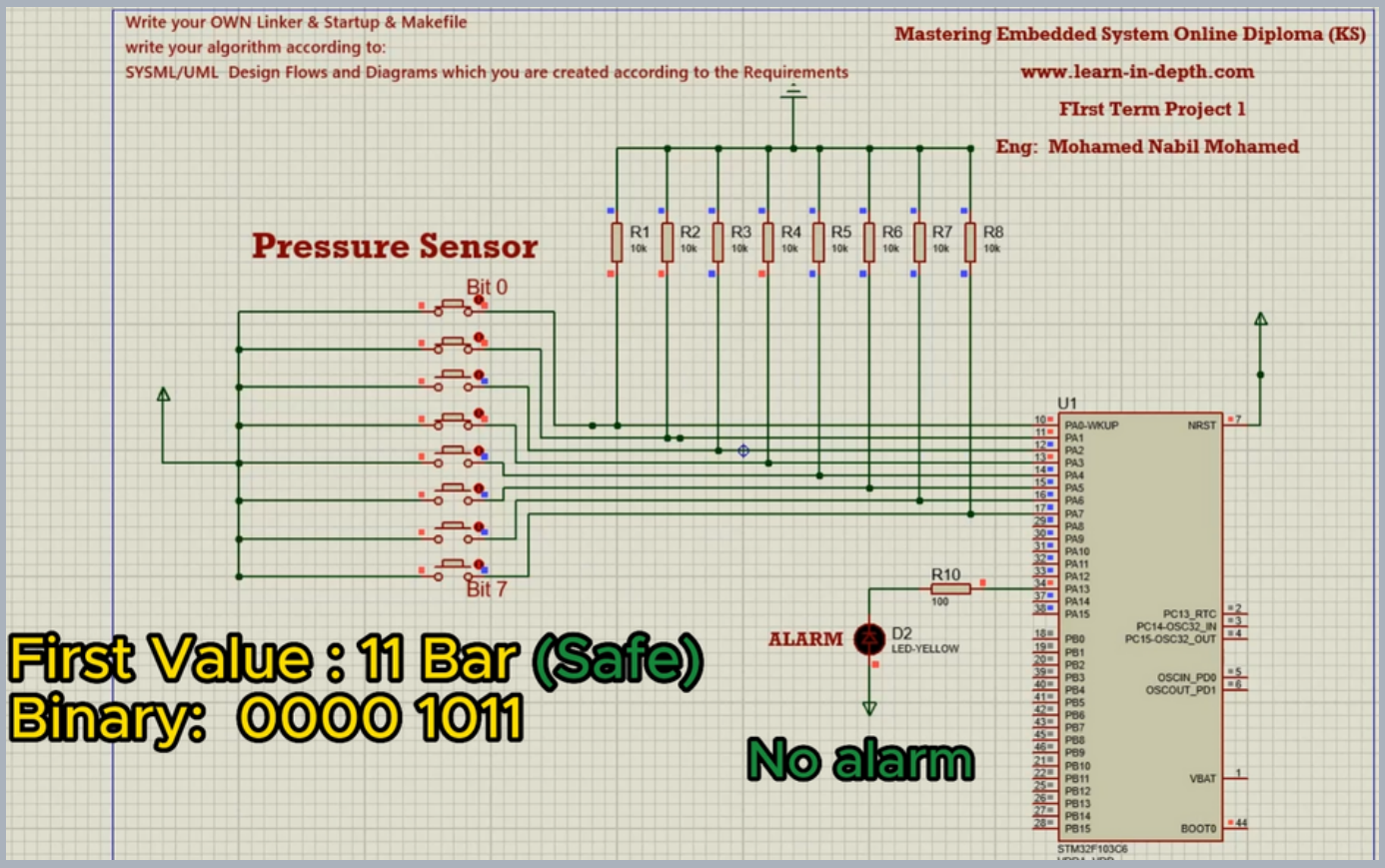
System Design

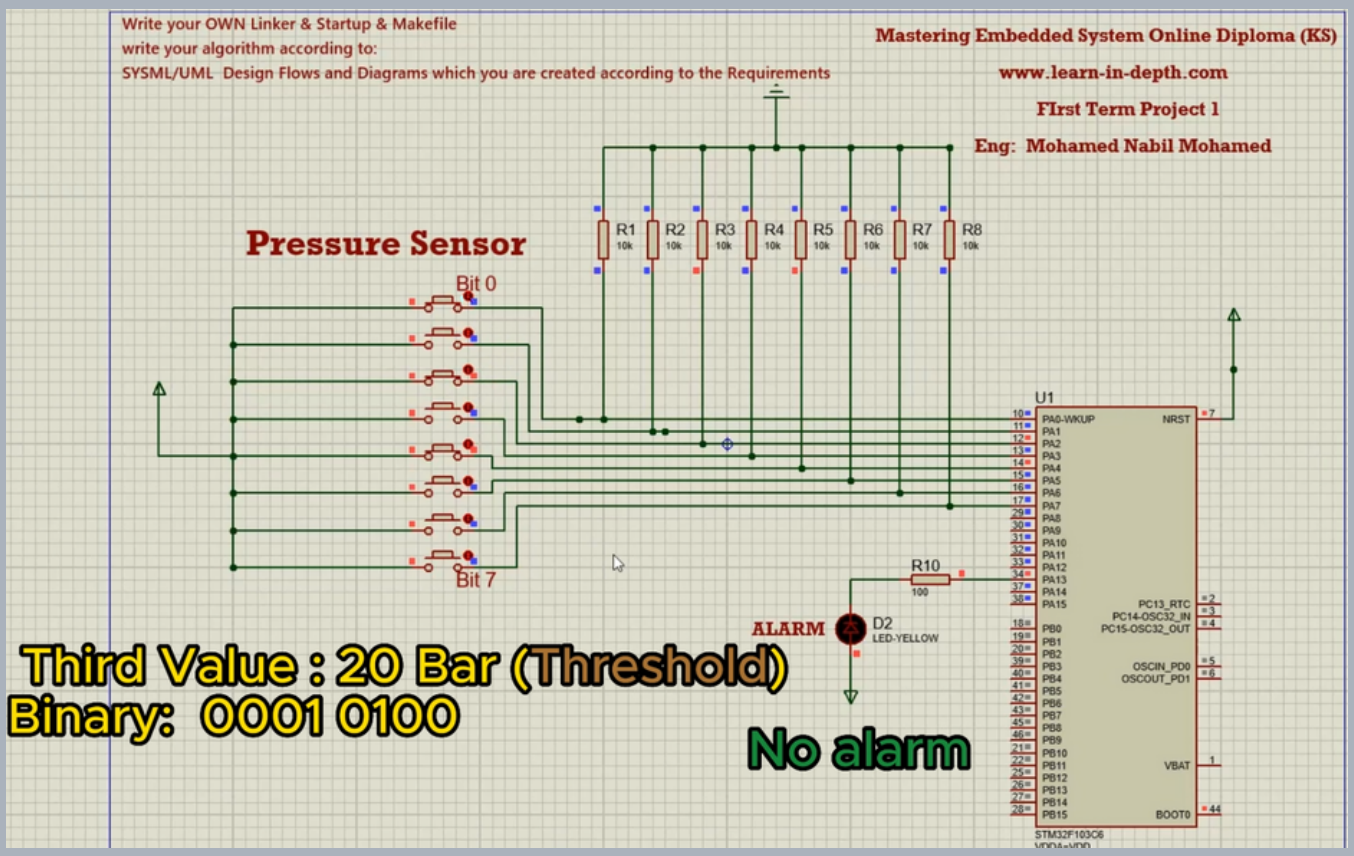
Alarm Actuator Driver State Machine



System Design

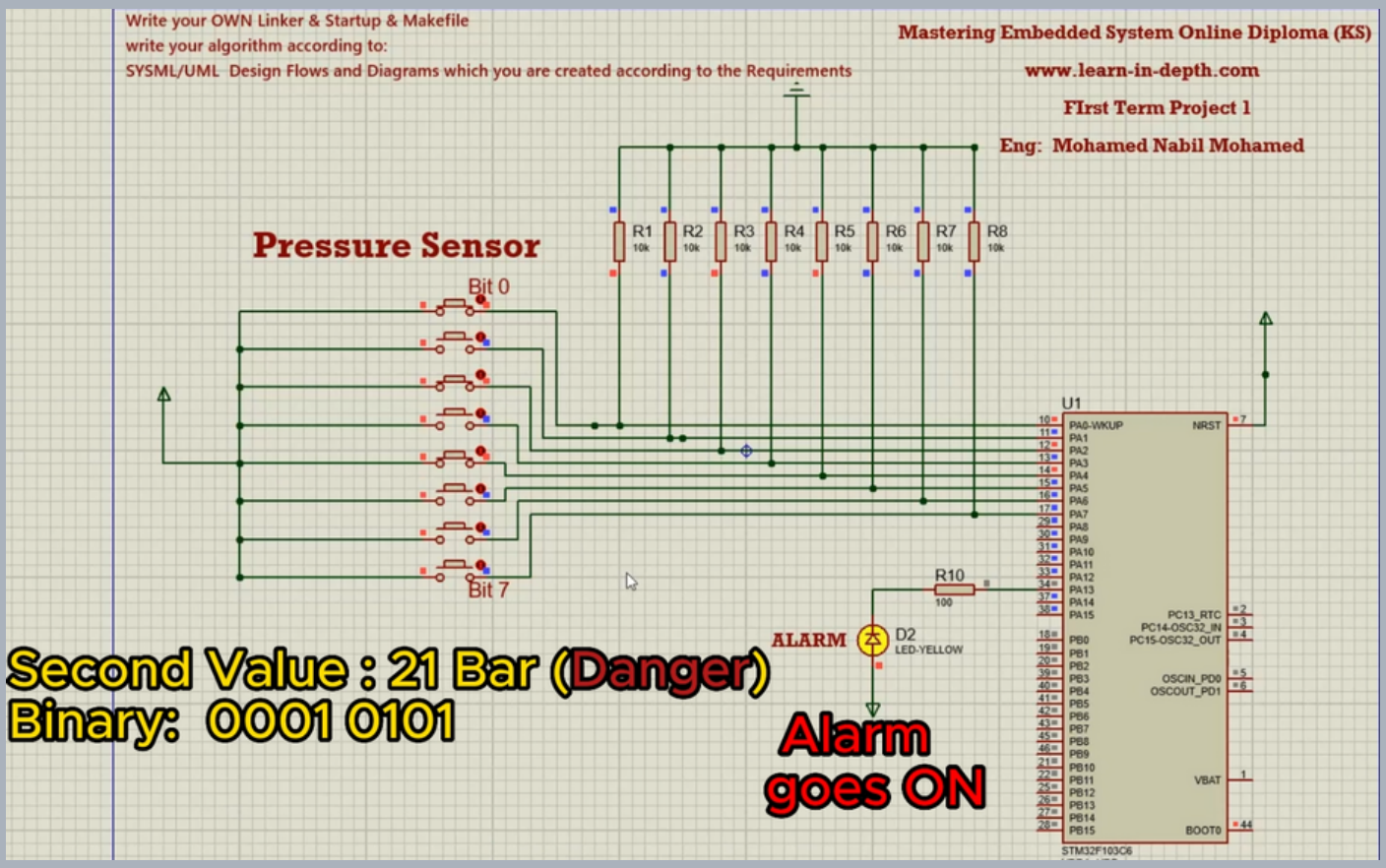
Simulation Results

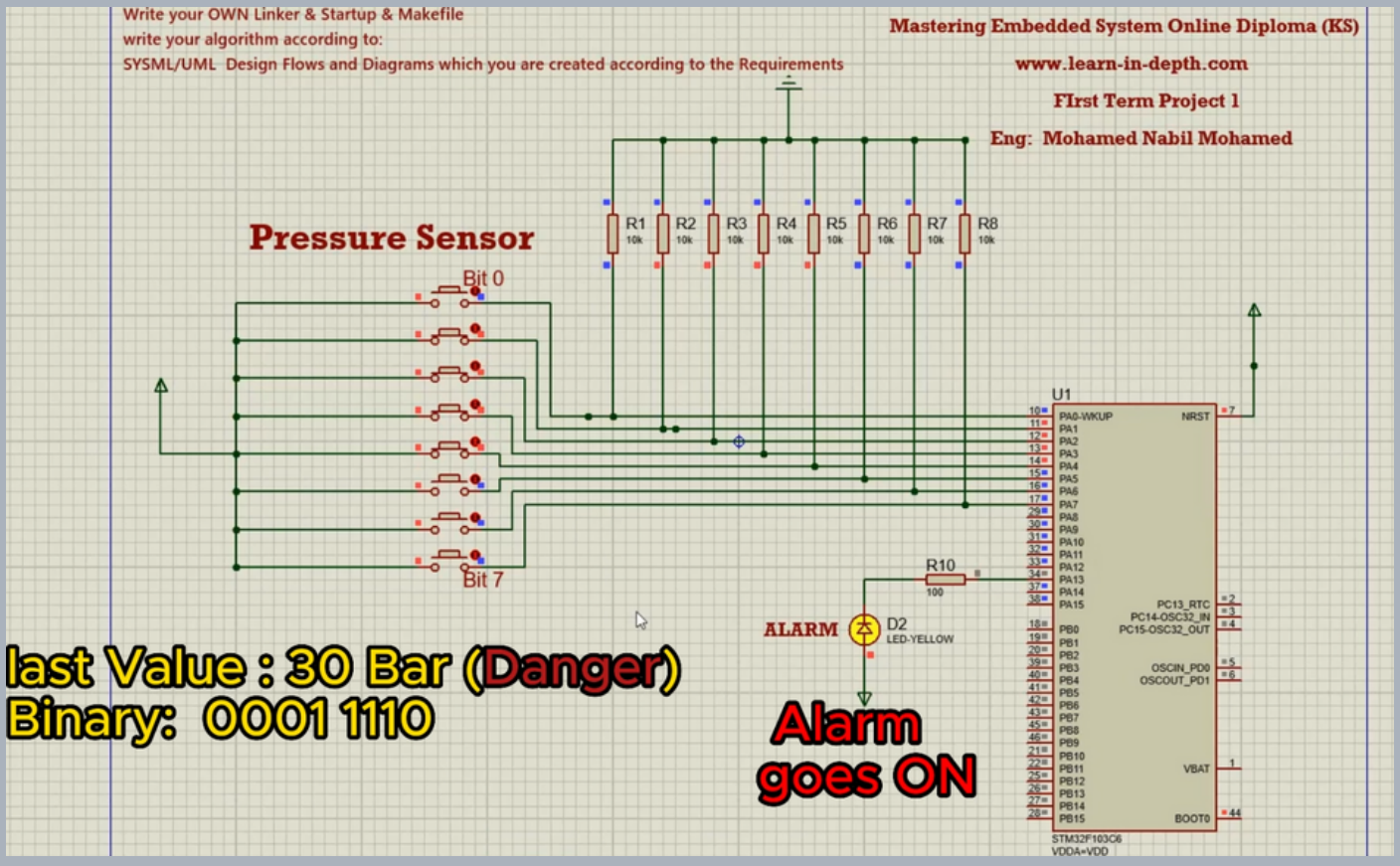




System Design

Simulation Results





System Design

Map File

```
Allocating common symbols
Common symbol       size              file

Alarm_Actuator_state_id
                    0x1               Alarm_Actuator_Driver.o
PSensor_state       0x4               main.o
Alarm_Actuator_state
                    0x4               Alarm_Actuator_Driver.o
Main_Algorithm_state
                    0x4               main.o
Main_Algorithm_state_id
                    0x1               main.o
PSensor_state_id    0x1               main.o
Alarm_Monitor_state_id
                    0x1               Alarm_Monitor.o
Alarm_Monitor_state
                    0x4               Alarm_Monitor.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      0x460
 *(.vector*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x0800001c       0x74 Alarm_Actuator_Driver.o
                0x0800001c                Alarm_Actuator_init
                0x08000038                Set_Alarm_Actuator
                0x08000054                Stop_Alarm_Actuator
                0x08000070                stateFunction_Alarm_Actuator_off
                0x08000080                stateFunction_Alarm_Actuator_on
 .text          0x08000090       0x94 Alarm_Monitor.o
                0x08000090                HighPressureDetected
                0x080000ac                stateFunction_Alarm_Monitor_AlarmOff
                0x080000c4                stateFunction_Alarm_Monitor_AlarmOn
                0x080000f0                stateFunction_Alarm_Monitor_waiting
 .text          0x08000124       0x10c driver.o
                0x08000124                Delay
                0x08000148                getPressureVal
                0x08000160                Set_Alarm_actuator
                0x080001b0                GPIO_INITIALIZATION
 .text          0x08000230        0x8c main.o
                0x08000230                setup
                0x08000268                main
 .text          0x080002bc        0x74 Main_Algorithm.o
                0x080002bc                SetPressureValue
                0x080002ec                stateFunction_High_Pressure_Check
 .text          0x08000330        0x74 Psensor_Driver.o
                0x08000330                PressureSensor_init
                0x0800034c                stateFunction_PSensor_busy
                0x08000384                stateFunction_PSensor_idle
 .text          0x080003a4        0xbc startup.o
                0x080003a4                Reset_Handler
                0x08000454                MM_Fault_Handler
                0x08000454                H_Fault_Handler
                0x08000454                Bus_Fault
                0x08000454                Default_Handler
                0x08000454                Usage_Fault_Handler
                0x08000454                NMI_Handler
 *(.rodata)
                0x08000460                _E_text = .

.glue_7         0x08000460          0x0
 .glue_7        0x00000000          0x0 linker stubs

.glue_7t        0x08000460          0x0
 .glue_7t       0x00000000          0x0 linker stubs

.vfp11_veneer   0x08000460          0x0
 .vfp11_veneer  0x00000000          0x0 linker stubs

```

System Design

Map File

```
.v4_bx          0x08000460        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x08000460        0x0
 .iplt          0x00000000        0x0 Alarm_Actuator_Driver.o

.rel.dyn        0x08000460        0x0
 .rel.iplt      0x00000000        0x0 Alarm_Actuator_Driver.o

.data           0x20000000        0x4 load address 0x08000460
                0x20000000                _S_data = .
 *(.data)
 .data          0x20000000        0x0 Alarm_Actuator_Driver.o
 .data          0x20000000        0x0 Alarm_Monitor.o
 .data          0x20000000        0x0 driver.o
 .data          0x20000000        0x0 main.o
 .data          0x20000000        0x4 Main_Algorithm.o
                0x20000000                Main_pressure_threshold
 .data          0x20000004        0x0 Psensor_Driver.o
 .data          0x20000004        0x0 startup.o
                0x20000004                . = ALIGN (0x4)
                0x20000004                _E_data = .

.igot.plt       0x20000004        0x0 load address 0x08000464
 .igot.plt      0x00000000        0x0 Alarm_Actuator_Driver.o

.bss            0x20000004     0x1022 load address 0x08000464
                0x20000004                _S_bss = .
 *(.bss)
 .bss           0x20000004        0x0 Alarm_Actuator_Driver.o
 .bss           0x20000004        0x0 Alarm_Monitor.o
 .bss           0x20000004        0x0 driver.o
 .bss           0x20000004        0x0 main.o
 .bss           0x20000004        0x4 Main_Algorithm.o
                0x20000004                Main_pressureVal
 .bss           0x20000008        0x4 Psensor_Driver.o
                0x20000008                Psensor_pressureVal
 .bss           0x2000000c        0x0 startup.o
                0x2000000c                _E_bss = .
                0x2000000c                . = ALIGN (0x4)
              0x2000100c                . = (. + 0x1000)
 *fill*         0x2000000c     0x1000
                0x2000100c                _stack_top = .
 COMMON         0x2000100c        0x8 Alarm_Actuator_Driver.o
                0x2000100c                Alarm_Actuator_state_id
                0x20001010                Alarm_Actuator_state
 COMMON         0x20001014        0x8 Alarm_Monitor.o
                0x20001014                Alarm_Monitor_state_id
                0x20001018                Alarm_Monitor_state
 COMMON         0x2000101c        0xa main.o
                0x2000101c                PSensor_state
                0x20001020                Main_Algorithm_state
                0x20001024                Main_Algorithm_state_id
                0x20001025                PSensor_state_id
LOAD Alarm_Actuator_Driver.o
LOAD Alarm_Monitor.o
LOAD driver.o
LOAD main.o
LOAD Main_Algorithm.o
LOAD Psensor_Driver.o
LOAD startup.o
OUTPUT(High_Pressure_Detection_System.elf elf32-littlearm)

.debug_info     0x00000000      0x866
 .debug_info    0x00000000      0x129 Alarm_Actuator_Driver.o
 .debug_info    0x00000129      0x11a Alarm_Monitor.o
 .debug_info    0x00000243      0x103 driver.o
 .debug_info    0x00000346      0x199 main.o
 .debug_info    0x000004df      0x10e Main_Algorithm.o
 .debug_info    0x000005ed      0x111 Psensor_Driver.o
 .debug_info    0x000006fe      0x168 startup.o

.debug_abbrev   0x00000000      0x44e
 .debug_abbrev  0x00000000       0x92 Alarm_Actuator_Driver.o
 .debug_abbrev  0x00000092       0x92 Alarm_Monitor.o
 .debug_abbrev  0x00000124       0x9d driver.o
 .debug_abbrev  0x000001c1       0x94 main.o
 .debug_abbrev  0x00000255       0xa5 Main_Algorithm.o
 .debug_abbrev  0x000002fa       0x92 Psensor_Driver.o
 .debug_abbrev  0x0000038c       0xc2 startup.o
```

System Design

Symbols Table

```
$ arm-none-eabi-nm.exe High_Pressure_Detection_System.elf
2000000c B _E_bss
20000004 D _E_data
08000460 T _E_text
20000004 B _S_bss
20000000 D _S_data
2000100c B _stack_top
0800001c T Alarm_Actuator_init
20001010 B Alarm_Actuator_state
2000100c B Alarm_Actuator_state_id
20001018 B Alarm_Monitor_state
20001014 B Alarm_Monitor_state_id
08000454 W Bus_Fault
08000454 T Default_Handler
08000124 T Delay
08000148 T getPressureVal
080001b0 T GPIO_INITIALIZATION
08000454 W H_Fault_Handler
08000090 T HighPressureDetected
08000268 T main
20001020 B Main_Algorithm_state
20001024 B Main_Algorithm_state_id
20000000 D Main_pressure_threshold
20000004 B Main_pressureVal
08000454 W MM_Fault_Handler
08000454 W NMI_Handler
08000330 T PressureSensor_init
20000008 B Psensor_pressureVal
2000101c B PSensor_state
20001025 B PSensor_state_id
080003a4 T Reset_Handler
08000160 T Set_Alarm_actuator
08000038 T Set_Alarm_Actuator
080002bc T SetPressureValue
08000230 T setup
08000070 T stateFunction_Alarm_Actuator_off
08000080 T stateFunction_Alarm_Actuator_on
080000ac T stateFunction_Alarm_Monitor_AlarmOff
080000c4 T stateFunction_Alarm_Monitor_AlarmOn
080000f0 T stateFunction_Alarm_Monitor_waiting
080002ec T stateFunction_High_Pressure_Check
0800034c T stateFunction_PSensor_busy
08000384 T stateFunction_PSensor_idle
08000054 T Stop_Alarm_Actuator
08000454 W Usage_Fault_Handler
08000000 T vectors
```

System Design 20

Symbols Table:

```
$ arm-none-eabi-nm.exe High_Pressure_Detection_System.elf
2000000c B _E_bss
20000004 D _E_data
08000460 T _E_text
20000004 B _S_bss
20000000 D _S_data
2000100c B _stack_top
0800001c T Alarm_Actuator_init
20001010 B Alarm_Actuator_state
2000100c B Alarm_Actuator_state_id
20001018 B Alarm_Monitor_state
20001014 B Alarm_Monitor_state_id
08000454 W Bus_Fault
08000454 T Default_Handler
08000124 T Delay
08000148 T getPressureVal
080001b0 T GPIO_INITIALIZATION
08000454 W H_Fault_Handler
08000090 T HighPressureDetected
08000268 T main
20001020 B Main_Algorithm_state
20001024 B Main_Algorithm_state_id
20000000 D Main_pressure_threshold
20000004 B Main_pressureVal
08000454 W MM_Fault_Handler
08000454 W NMI_Handler
08000330 T PressureSensor_init
20000008 B Psensor_pressureVal
2000101c B PSensor_state
20001025 B PSensor_state_id
080003a4 T Reset_Handler
08000160 T Set_Alarm_actuator
08000038 T Set_Alarm_Actuator
080002bc T SetPressureValue
08000230 T setup
08000070 T stateFunction_Alarm_Actuator_off
08000080 T stateFunction_Alarm_Actuator_on
080000ac T stateFunction_Alarm_Monitor_AlarmOff
080000c4 T stateFunction_Alarm_Monitor_AlarmOn
080000f0 T stateFunction_Alarm_Monitor_waiting
080002ec T stateFunction_High_Pressure_Check
0800034c T stateFunction_PSensor_busy
08000384 T stateFunction_PSensor_idle
08000054 T Stop_Alarm_Actuator
08000454 W Usage_Fault_Handler
08000000 T vectors
```

Section Table:

```
$ arm-none-eabi-objdump.exe -h High_Pressure_Detection_System.elf

High_Pressure_Detection_System.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000460  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  20000000  08000460  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001022  20000004  08000464  00010004  2**2
                  ALLOC
  3 .debug_info   00000866  00000000  00000000  00010004  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 0000044e  00000000  00000000  0001086a  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000003f8  00000000  00000000  00010cb8  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000e0  00000000  00000000  000110b0  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   0000038c  00000000  00000000  00011190  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    00000498  00000000  00000000  0001151c  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  000119b4  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000119c5  2**0
                  CONTENTS, READONLY
 11 .debug_frame  000002bc  00000000  00000000  000119f8  2**2
```