Pressure System

Customer requirement

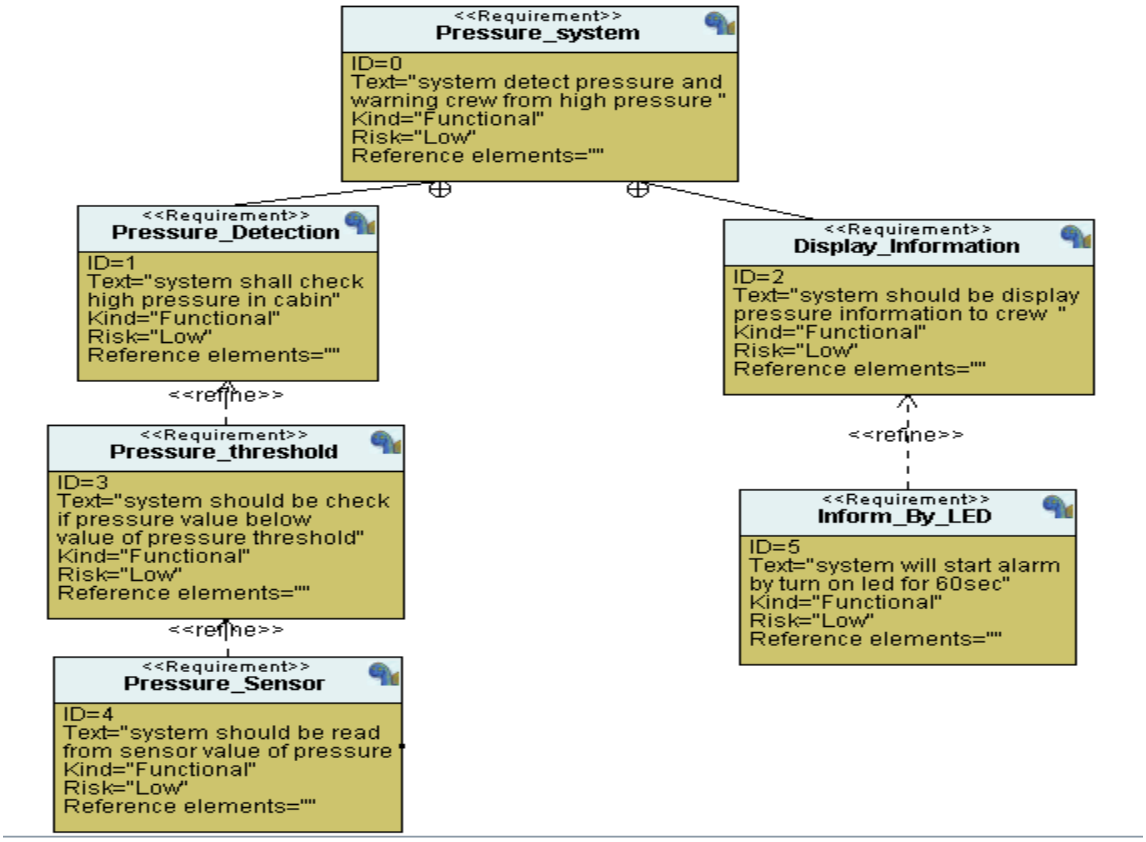
Specification:

1. System read pressure value from sensor in cabin.
2. System inform crew if pressure value above 20 bars by turn on led for 60 secs.

Assumptions:

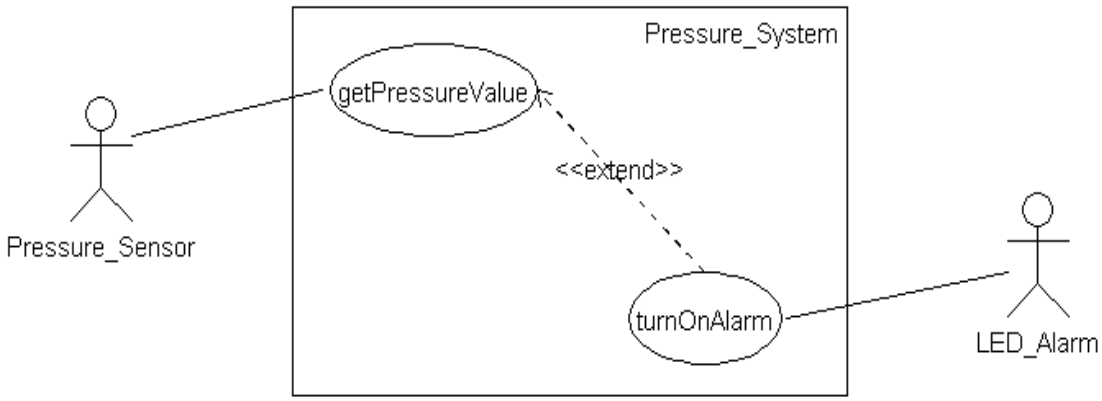
1. The pressure sensor never fails.
2. The led alarm never fails.

Customer requirement diagram:



Pressure system need two component, pressure detection and display information.

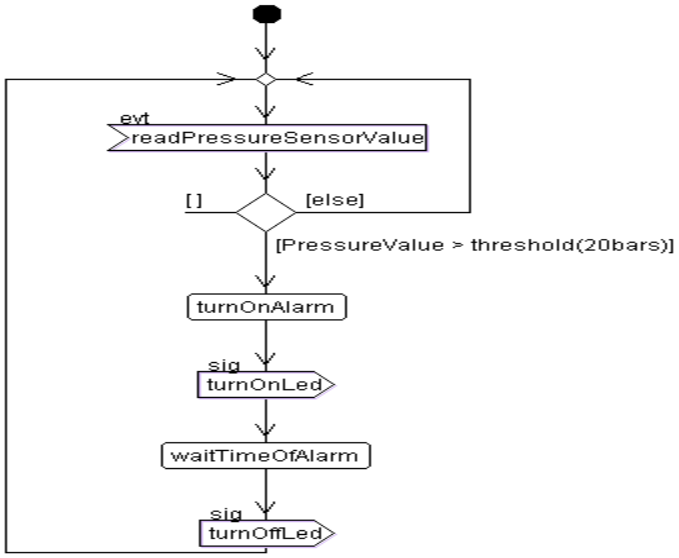
UML: Use Case Diagram



1. getPressureValue(Use case) get value of pressure from pressure_sensor(Actor).
2. turnOnAlaram(Use Case) turn on LED_Alarm if getPressureValue(Use case) get value of pressure larger than 20 bars.

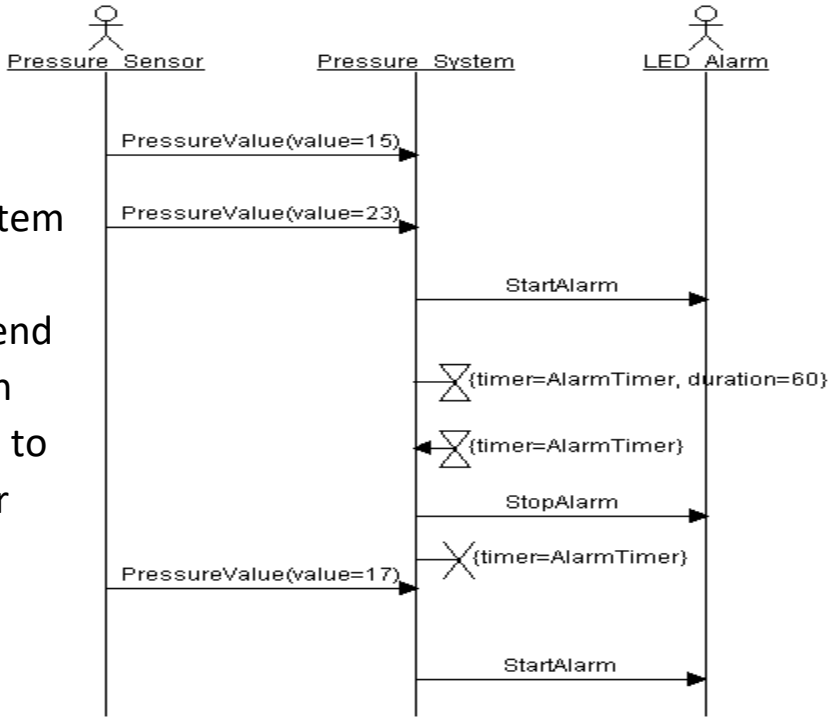
UML: Activity Diagram

1. start program
2. Read pressure value from sensor
3. If pressure value smaller than threshold return to waiting new value and check.
4. If pressure value larger than threshold will start alarm by turn on led for 60 sec then return to waiting new value and check.

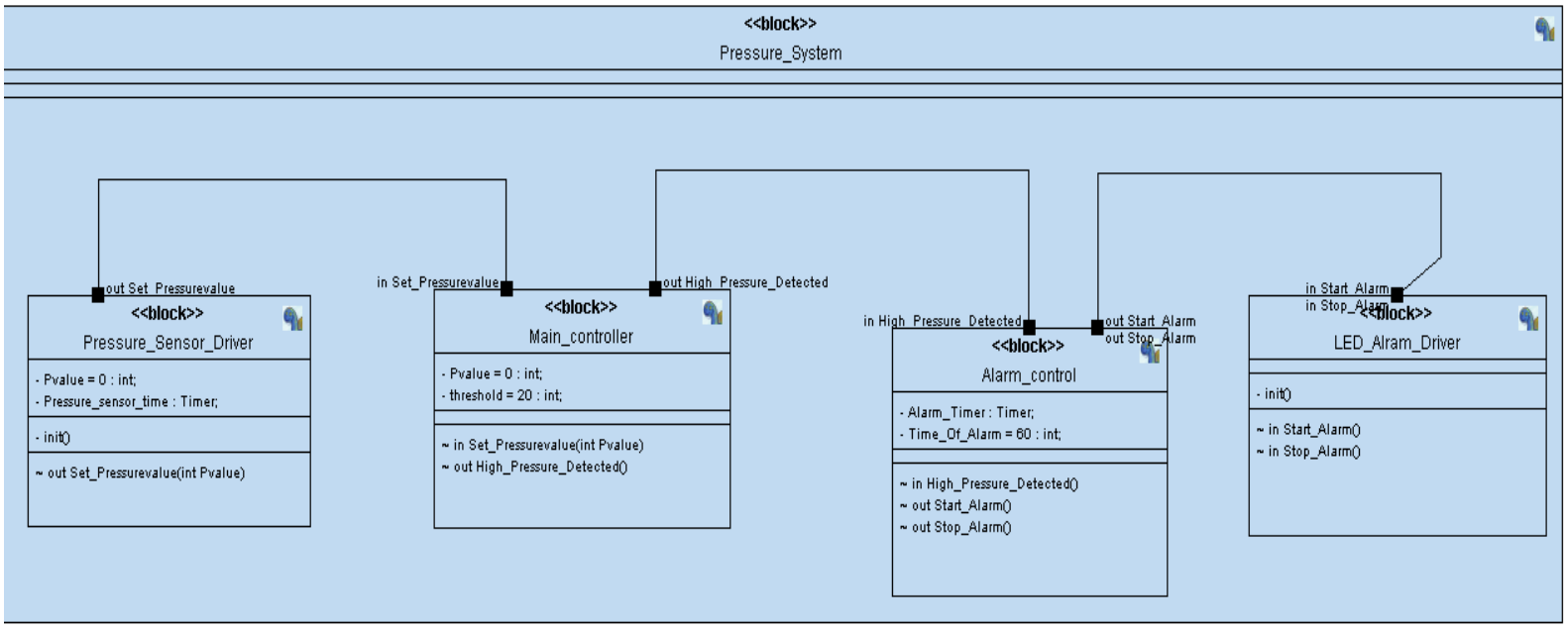


UML: Sequence Diagram

1. Assume pressure sensor send value to system smaller than threshold system won't happen any thing.
2. Assume pressure sensor send value to system larger than threshold system will send to led alarm to start alarm for 60 sec then stop alarm.

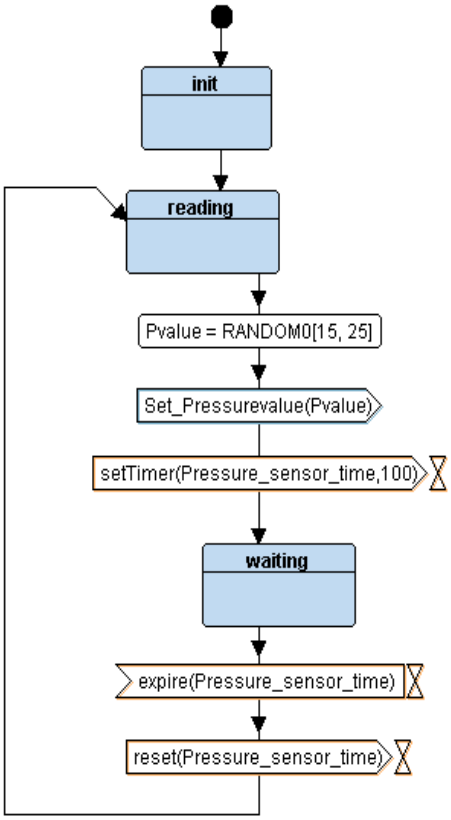


Design: Block Diagram



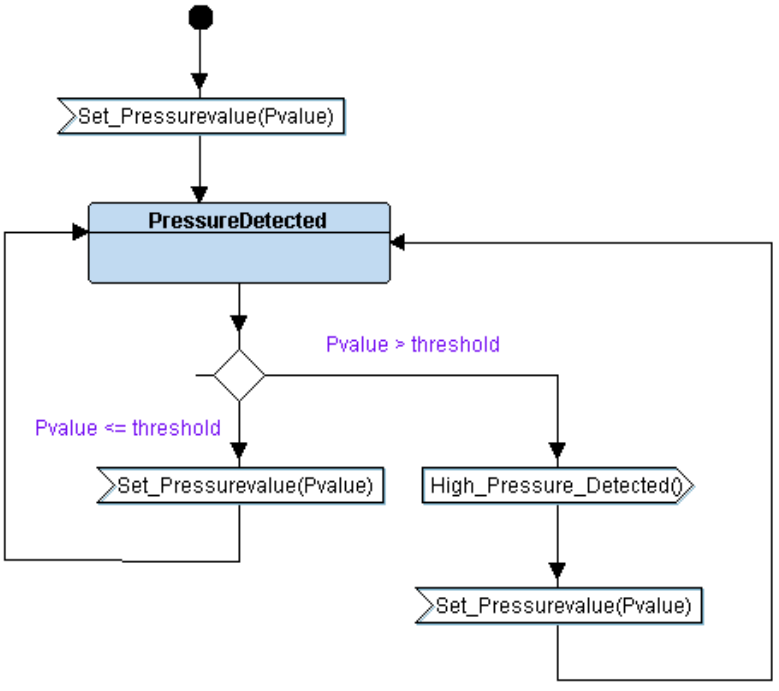
Block: Pressure Sensor Driver

1. Start will be initialized pressure sensor driver.
2. Go to reading state to get value of pressure.
3. Send pressure value to main controller.
4. wait for 100 sec to go to reading state again.



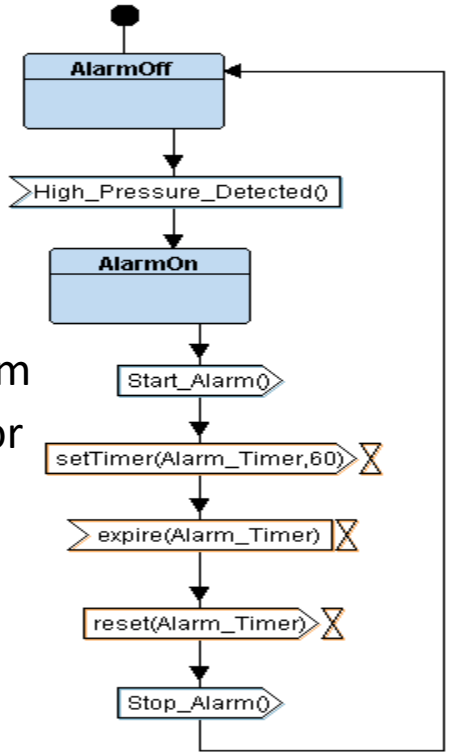
Block: Main Controller

1. Main controller wait to receive pressure value from pressure sensor
2. If pressure value smaller than threshold will be waiting to a new pressure value.
3. If pressure value smaller than threshold will be send signal to alarm control to be start alarm then wait to new pressure value.



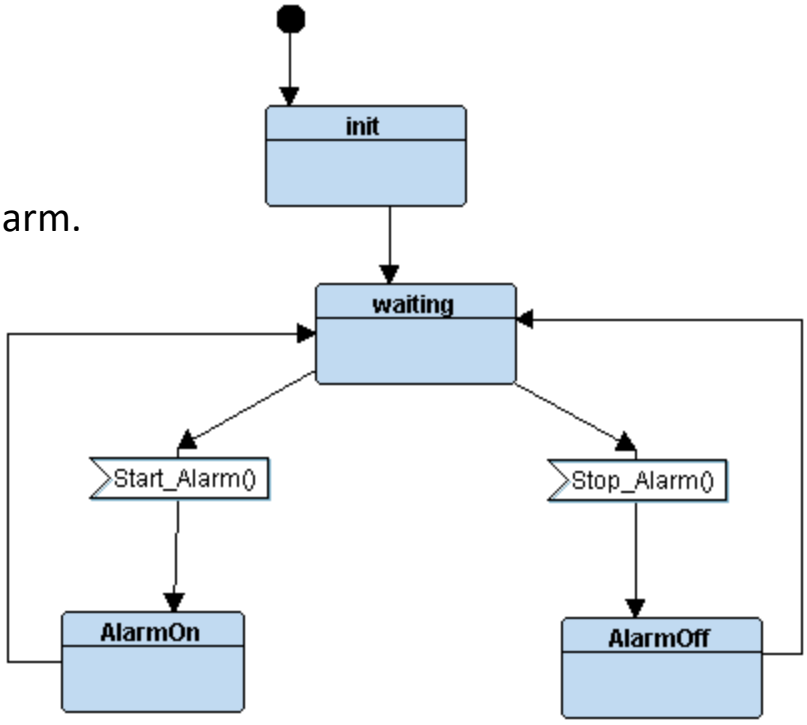
Block: Alarm controller

1. First alarm controller will be in state alarm off waiting to receive high pressure detected to go to state alarm on.
2. Alarm controller send to led_alarm signal to turn on then set timer for 60 sec then send signal to led_alarm to turn off then return to state alarm off.



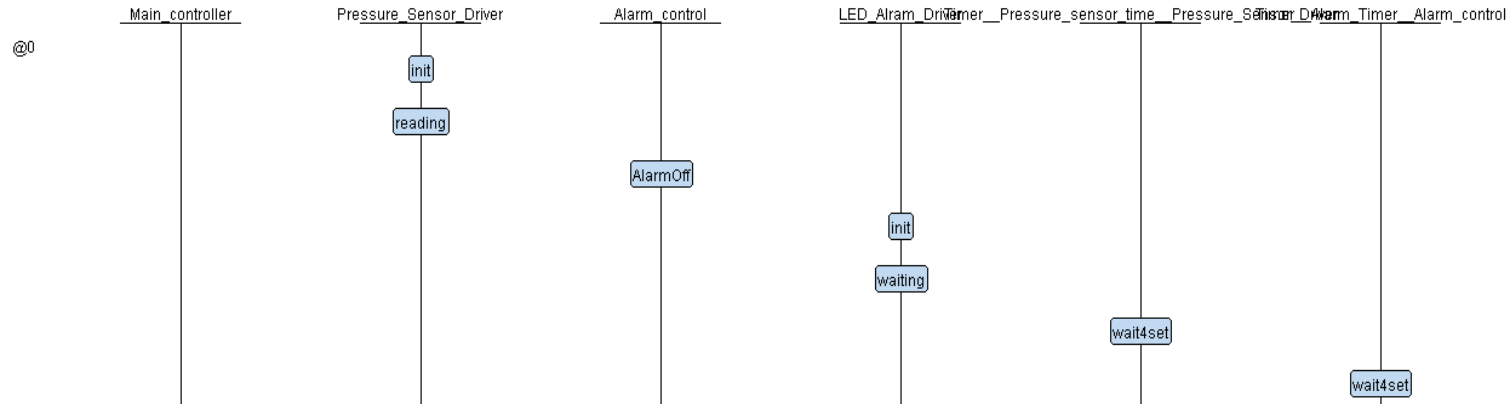
Block: LED Alarm Driver

1. Start will be initialized Led alarm driver.
2. Driver waiting to signal to start alarm or to stop alarm.
3. If signal start alarm will be go to state alarm on.
4. If signal stop alarm will be go to state alarm off.

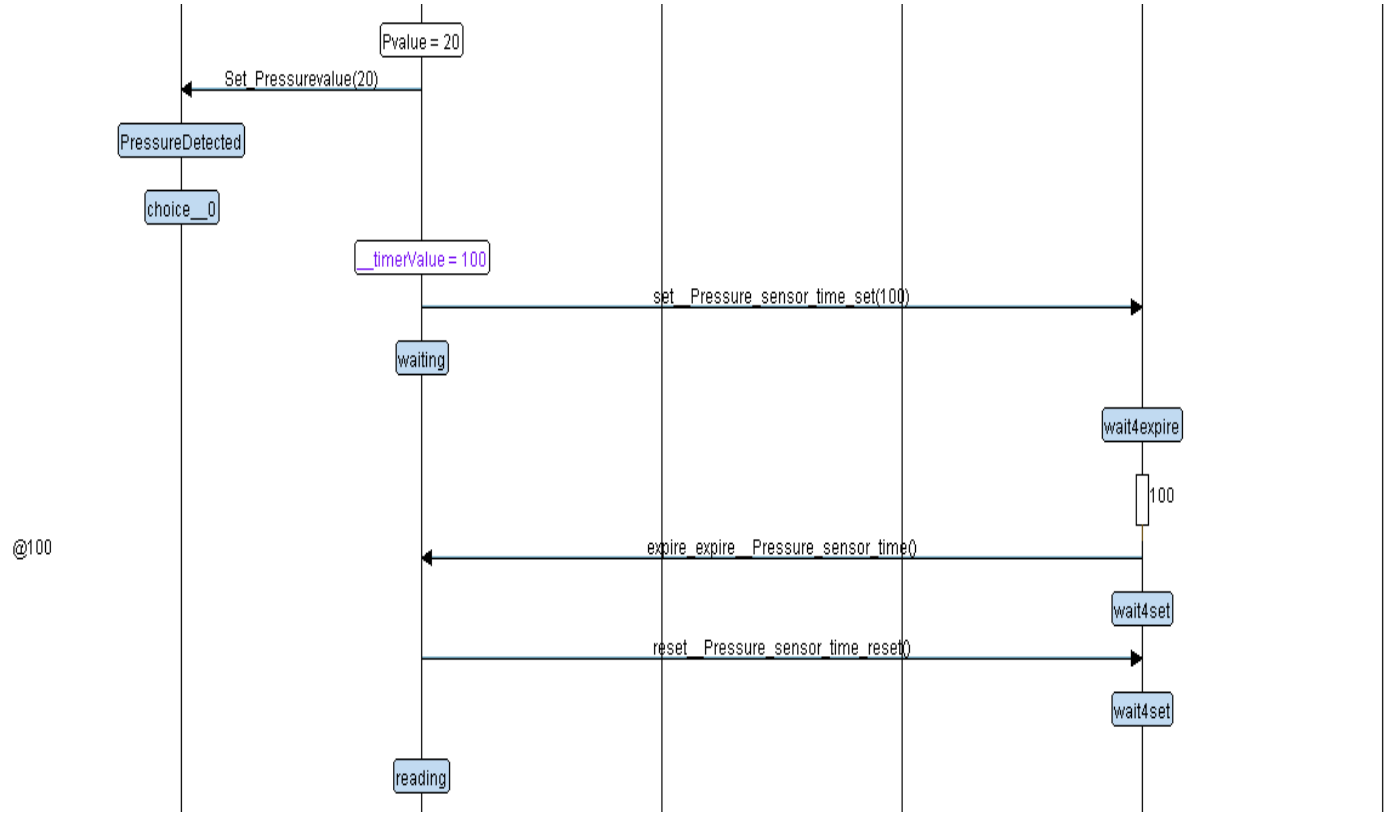


simulate project to show sequence

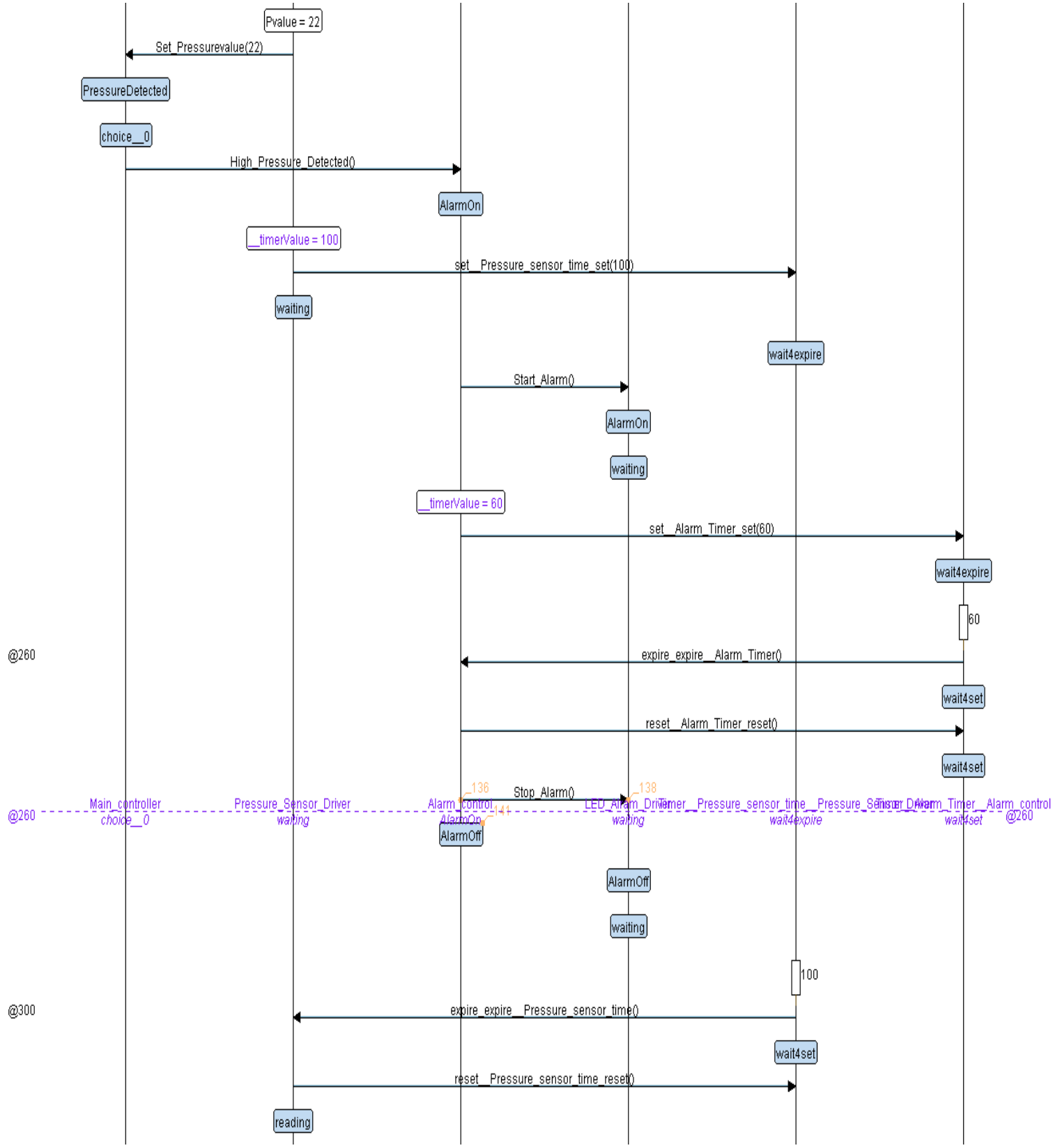
1. when start program.



2.when sensor read value smaller than threshold (20 bar).



3. when sensor read value bigger than threshold (20 bar).



After implement code will show symbols and sections for each file:

1. Symbols for driver of pressure sensor and Led alarm actuator:

```
Bolis@Bolis MINGW64 /e/Doc/Embedded System Diploma/project/Pressure_system
$ arm-none-eabi-nm.exe driver.o
00000000 T Delay
00000024 T getPressureVal
0000008c T GPIO_INITIALIZATION
0000003c T Set_Alarm_actuator
```

2. Symbols for Alarm control:

```
Bolis@Bolis MINGW64 /e/Doc/Embedded System Diploma/project/Pressure_system
$ arm-none-eabi-nm.exe Alarm_Control.o
         U Delay
00000000 T High_pressure_detected
         U Set_Alarm_actuator
00000000 D Time_Of_alarm
```

3. Symbols for main controller:

```
Bolis@Bolis MINGW64 /e/Doc/Embedded System Diploma/project/Pressure_system
$ arm-none-eabi-nm.exe main.o
         U Delay
         U getPressureVal
         U GPIO_INITIALIZATION
         U High_pressure_detected
00000000 T main
```

4. Symbols for Pressure system:

```
Bolis@Bolis MINGW64 /e/Doc/Embedded System Diploma/project/Pressure_system
$ arm-none-eabi-nm.exe Pressure_system.elf
08000190 t _reset
08000054 T Delay
08000078 T getPressureVal
080000e0 T GPIO_INITIALIZATION
0800001c T High_pressure_detected
08000160 T main
08000090 T Set_Alarm_actuator
08000198 D Time_Of_alarm
08000196 t Vector_handler
```

5. Sections for driver of pressure sensor and Led alarm actuator:

```
Bolis@Bolis MINGW64 /e/Doc/Embedded System Diploma/project/Pressure_system
$ arm-none-eabi-objdump.exe -h driver.o

driver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000010c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000140  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000140  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  00000140  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000033  00000000  00000000  00000152  2**0
                  CONTENTS, READONLY
```

6. Sections for Alarm control:

```
Bolis@Bolis MINGW64 /e/Doc/Embedded System Diploma/project/Pressure_system
$ arm-none-eabi-objdump.exe -h Alarm_Control.o

Alarm_Control.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000038  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000004  00000000  00000000  0000006c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000070  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  00000070  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000033  00000000  00000000  00000082  2**0
                  CONTENTS, READONLY
```

7. Sections for main controller:

```
Bolis@Bolis MINGW64 /e/Doc/Embedded System Diploma/project/Pressure_system
$ arm-none-eabi-objdump.exe -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000030  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000064  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000064  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  00000064  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000033  00000000  00000000  00000076  2**0
                  CONTENTS, READONLY
```

8. Symbols for Pressure system:

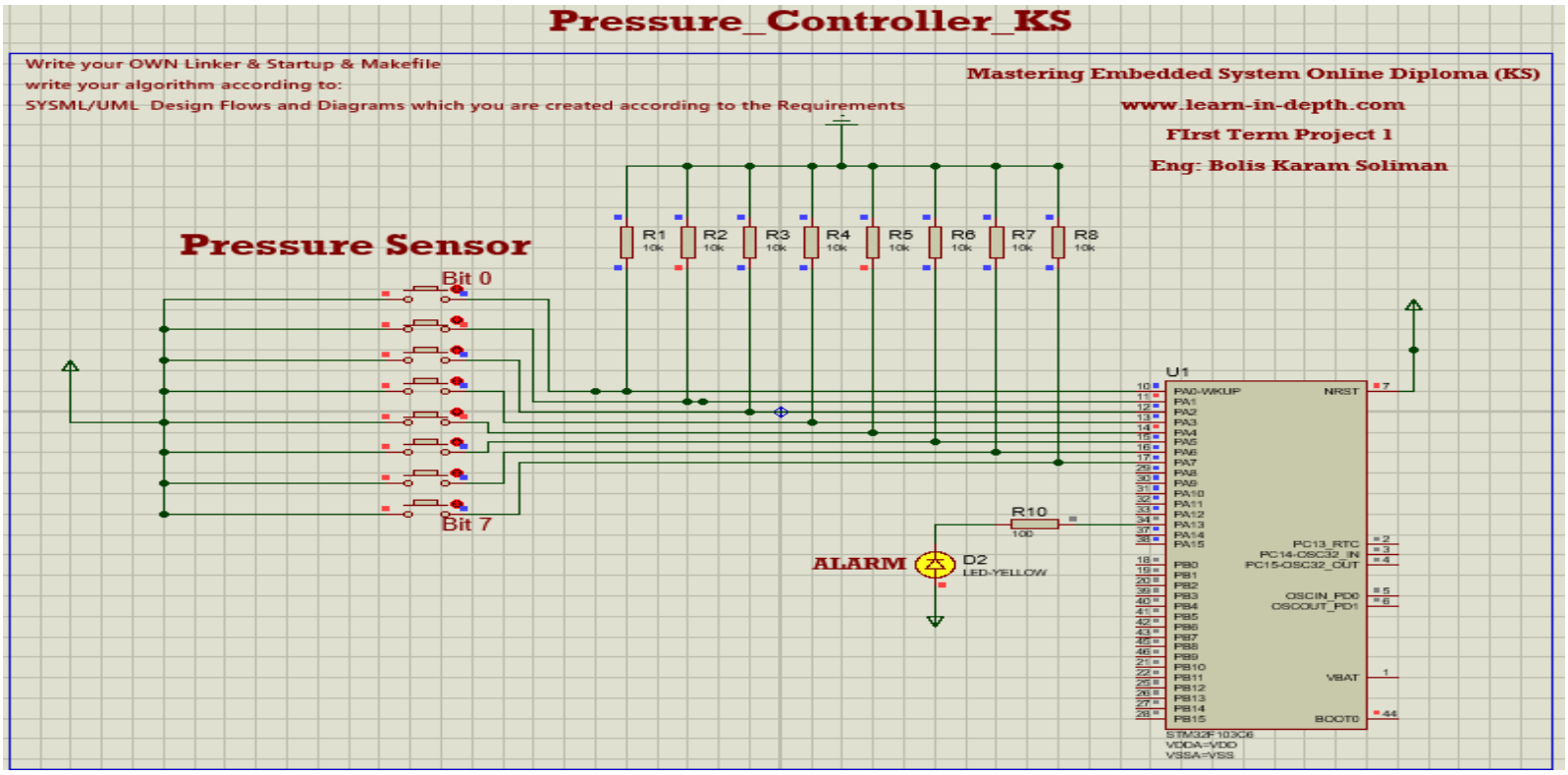
```
Bolis@Bolis MINGW64 /e/Doc/Embedded System Diploma/project/Pressure_system
$ arm-none-eabi-objdump.exe -h Pressure_system.elf

Pressure_system.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000198  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  08000198  08000198  00008198  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .comment      00000011  00000000  00000000  0000819c  2**0
                  CONTENTS, READONLY
  3 .ARM.attributes 00000031  00000000  00000000  000081ad  2**0
                  CONTENTS, READONLY
```

When run executable on simulation

1. When pressure value(18) smaller than threshold (20):



2. When pressure value(24) larger than threshold (20):