FIRST TERM PROJECT REPORT

Mastering Embedded System Online Diploma

www.learn-in-depth.com

First Term (Final Project 1: Pressure Controller)

Eng. Aya Ramadan Mohammed

My Profile:

https://www.learn-in-depth-store.com/profile/ayaramadanayaramadan22/profile

TABLE OF CONTENTS

Contents

1- Case Study..3

2- Requirements Diagram..4

3-System Analysis...5

1.Use Case Diagram...5

2.Activity Diagram...6

3.Sequence Diagram..7

4- System Design...8

5-Source Code..11

6- Testing...16

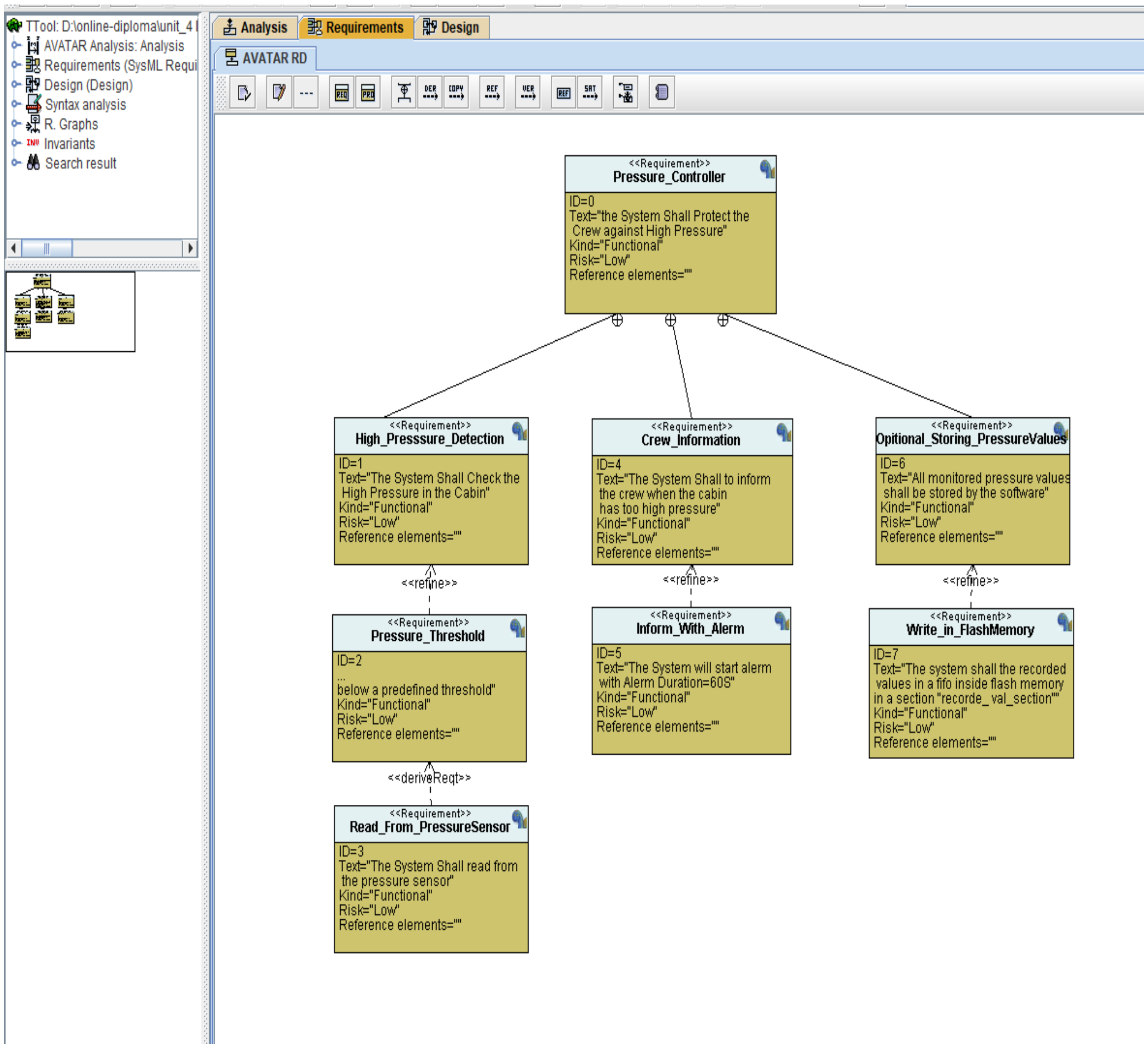
7-Simulation...17

Case Study (Pressure Controller)

Case Study : A pressure controller with an alarm the cabin

- **deliver the software of the following system:**
  - **Specification (from the client)**
    - pressure controller informs the crew of a cabin with an alarm when the pressure exceeds when the pressure exceeds 20 bars in the cabin
    - the alarm duration equals 60 seconds.

Requirements Diagram



<<Requirement>>
Pressure_Controller
ID=0
Text="the System Shall Protect the Crew against High Pressure"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
High_Presssure_Detection
ID=1
Text="The System Shall Check the High Pressure in the Cabin"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Crew_Information
ID=4
Text="The System Shall to inform the crew when the cabin has too high pressure"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Opitional_Storing_PressureValues
ID=6
Text="All monitored pressure values shall be stored by the software"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Pressure_Threshold
ID=2
...
below a predefined threshold"
Kind="Functional"
Risk="Low"
Reference elements=""

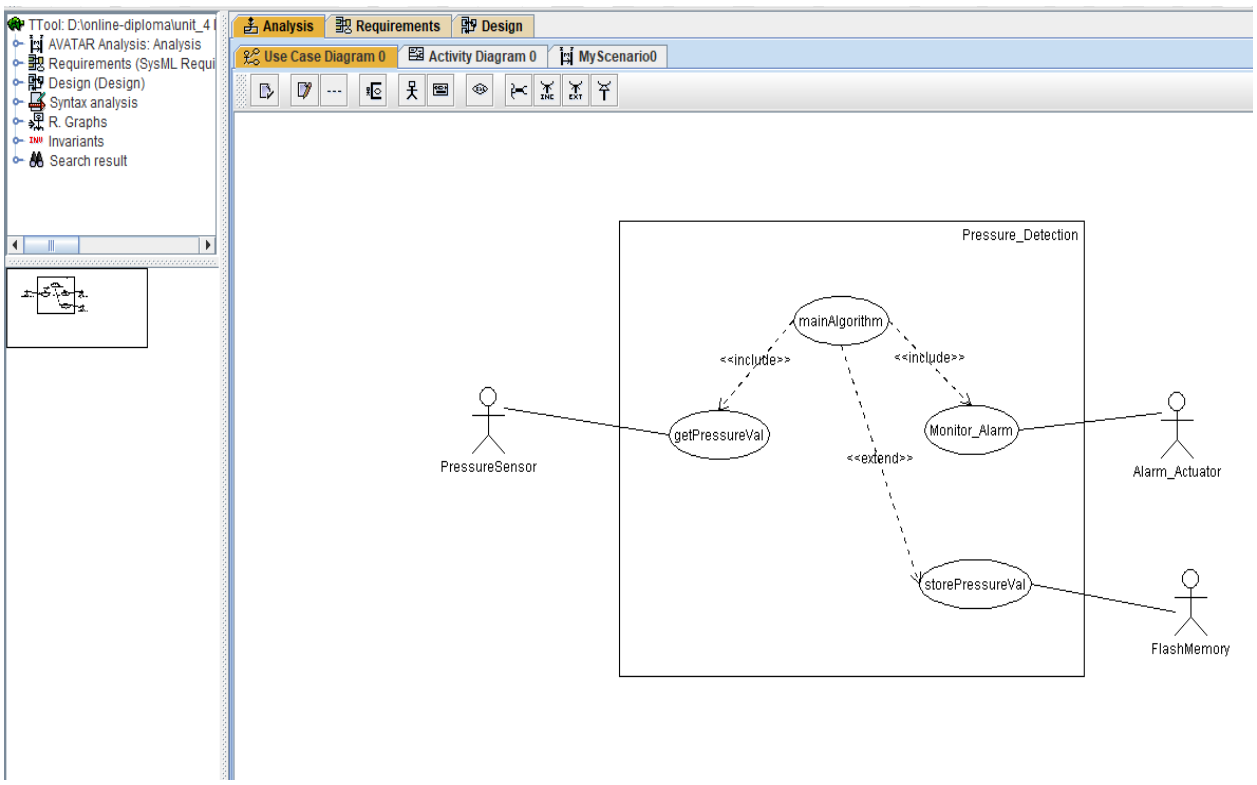
<<Requirement>>
Inform_With_Alerm
ID=5
Text="The System will start alerm with Alerm Duration=60S"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Write_in_FlashMemory
ID=7
Text="The system shall the recorded values in a fifo inside flash memory in a section "recorde_ val_section""
Kind="Functional"
Risk="Low"
Reference elements=""

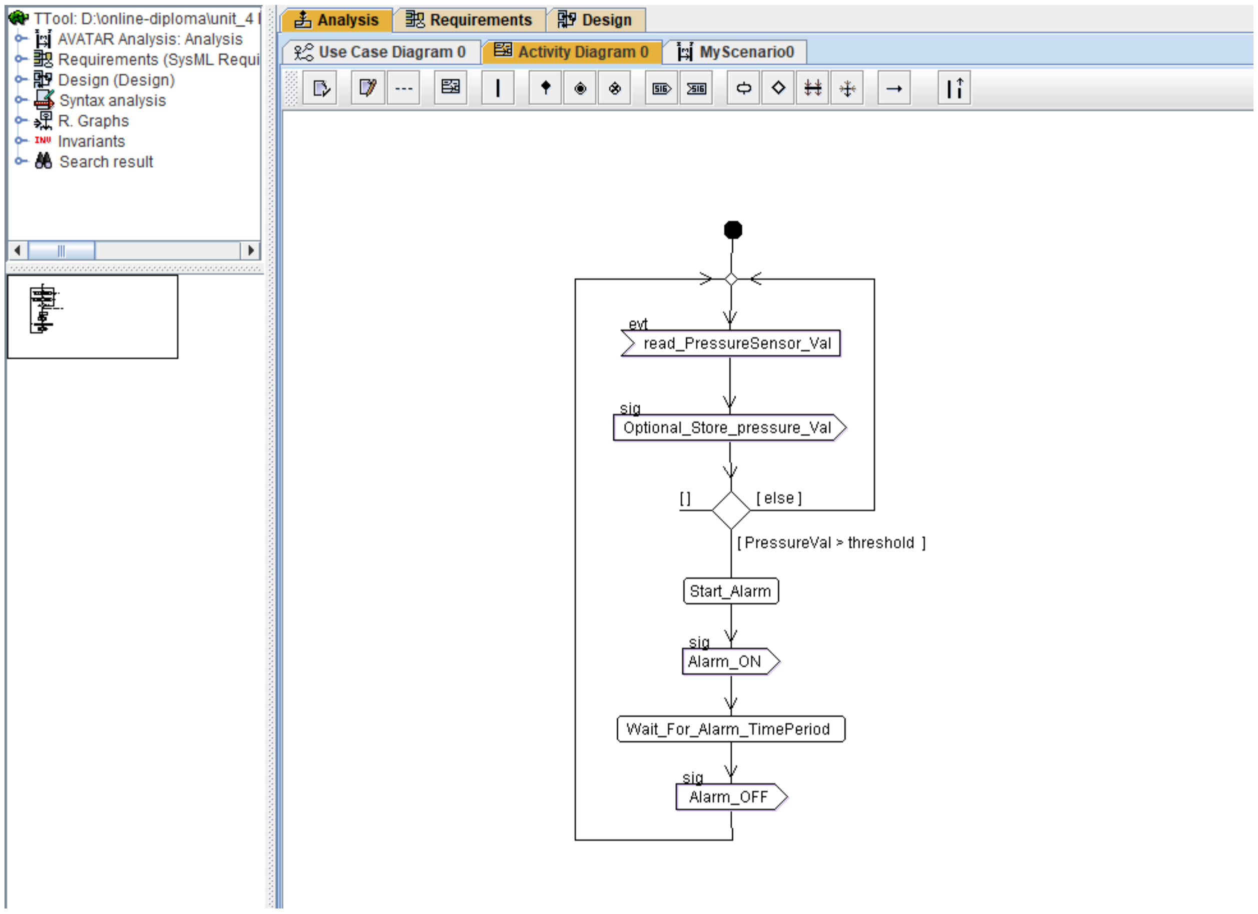
<<Requirement>>
Read_From_PressureSensor
ID=3
Text="The System Shall read from the pressure sensor"
Kind="Functional"
Risk="Low"
Reference elements=""

System Analysis

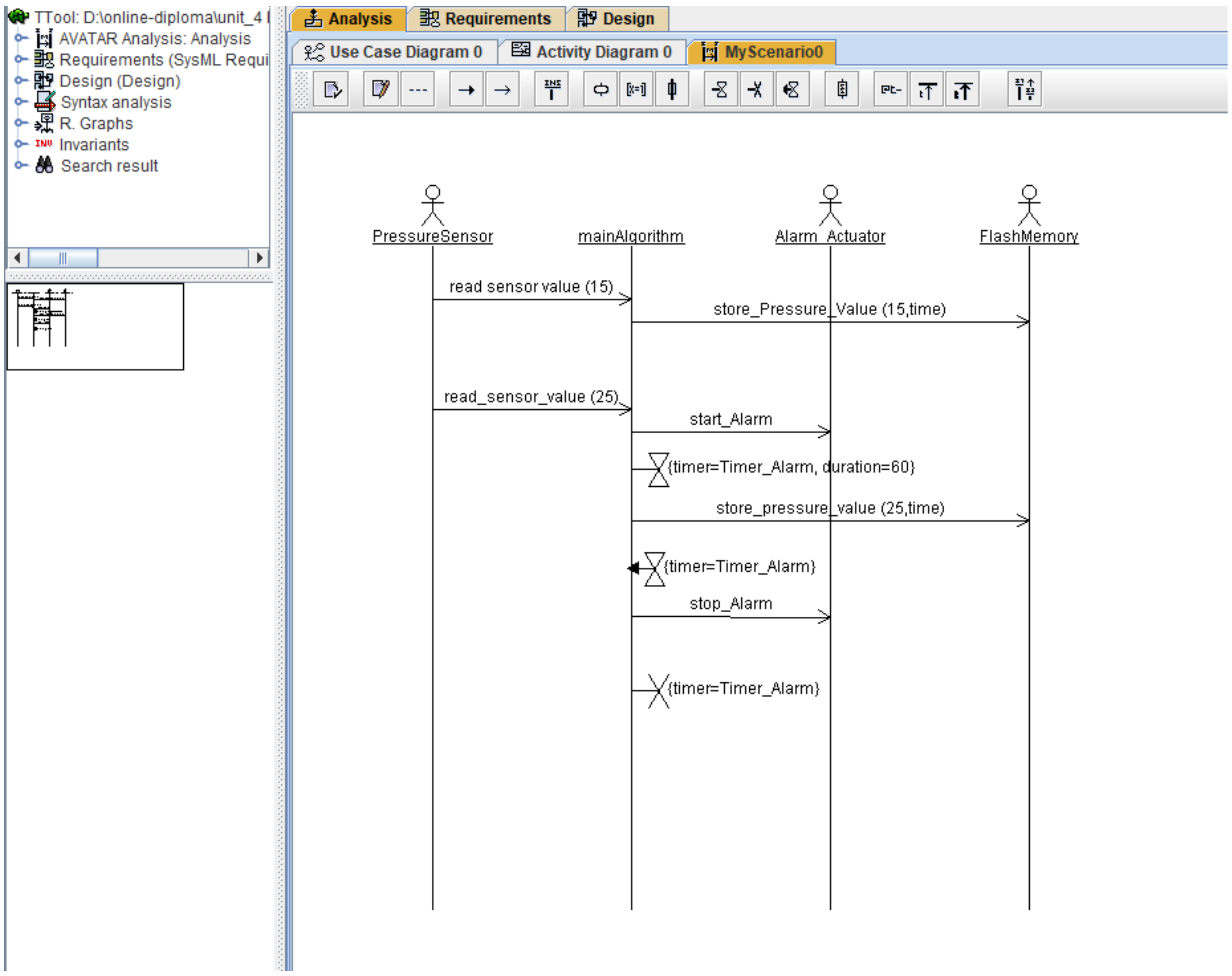
➢ Use Case Diagram



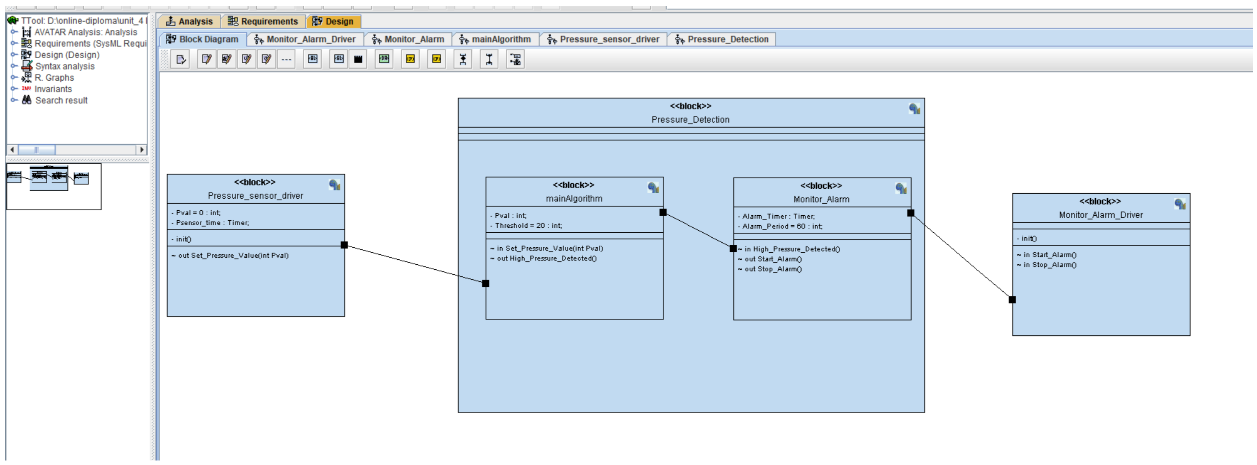
➢ Activity Diagram

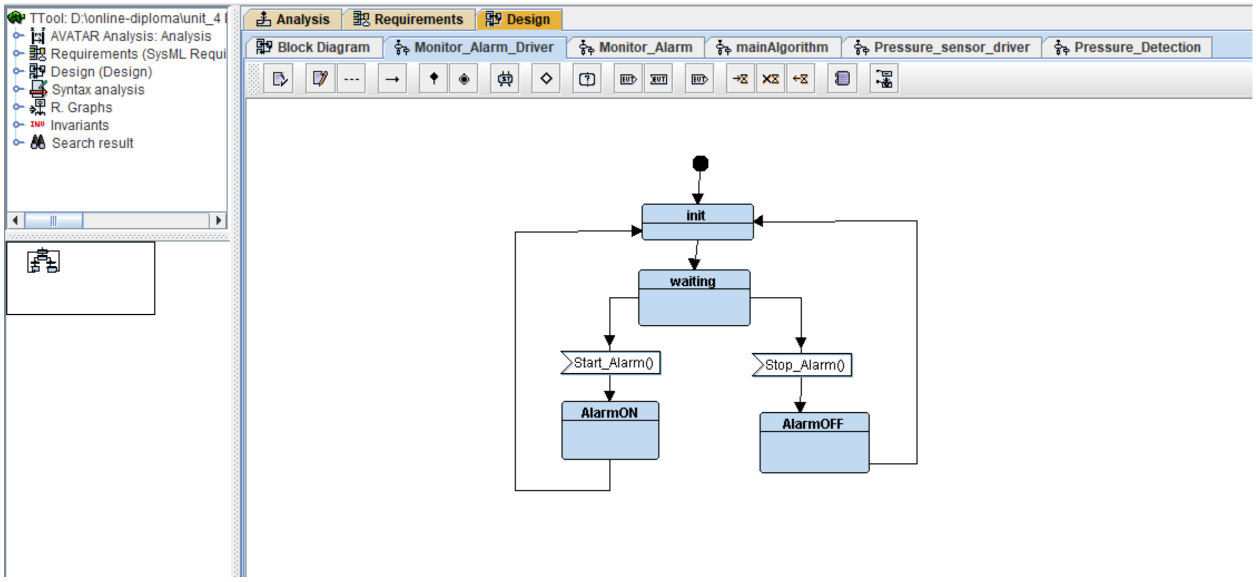


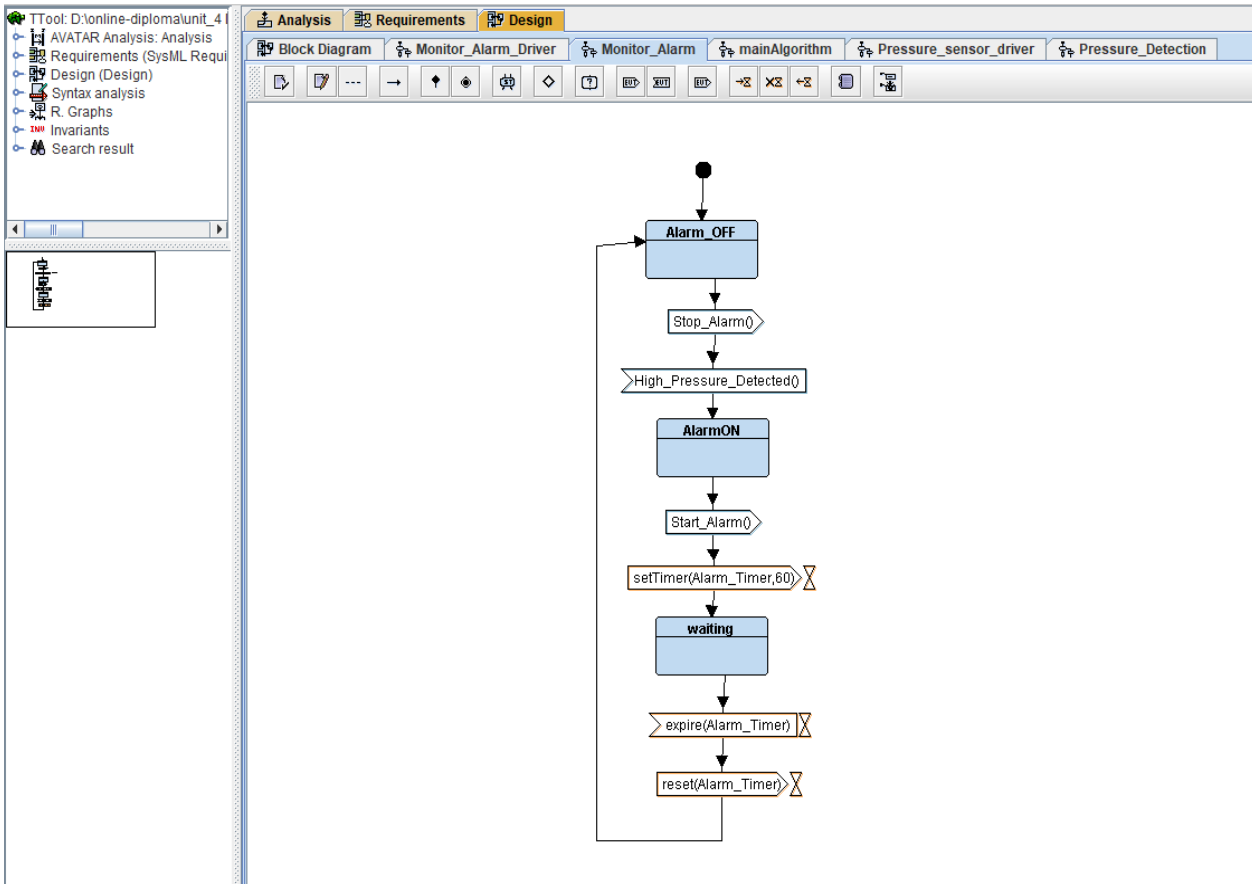
➢ Sequence Diagram

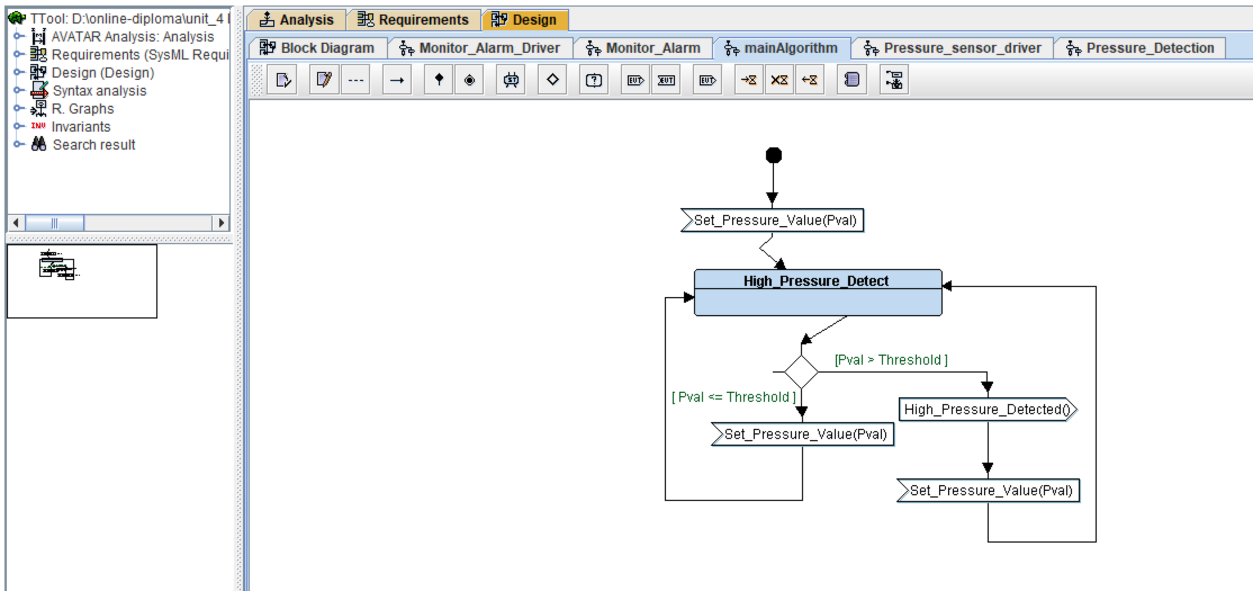


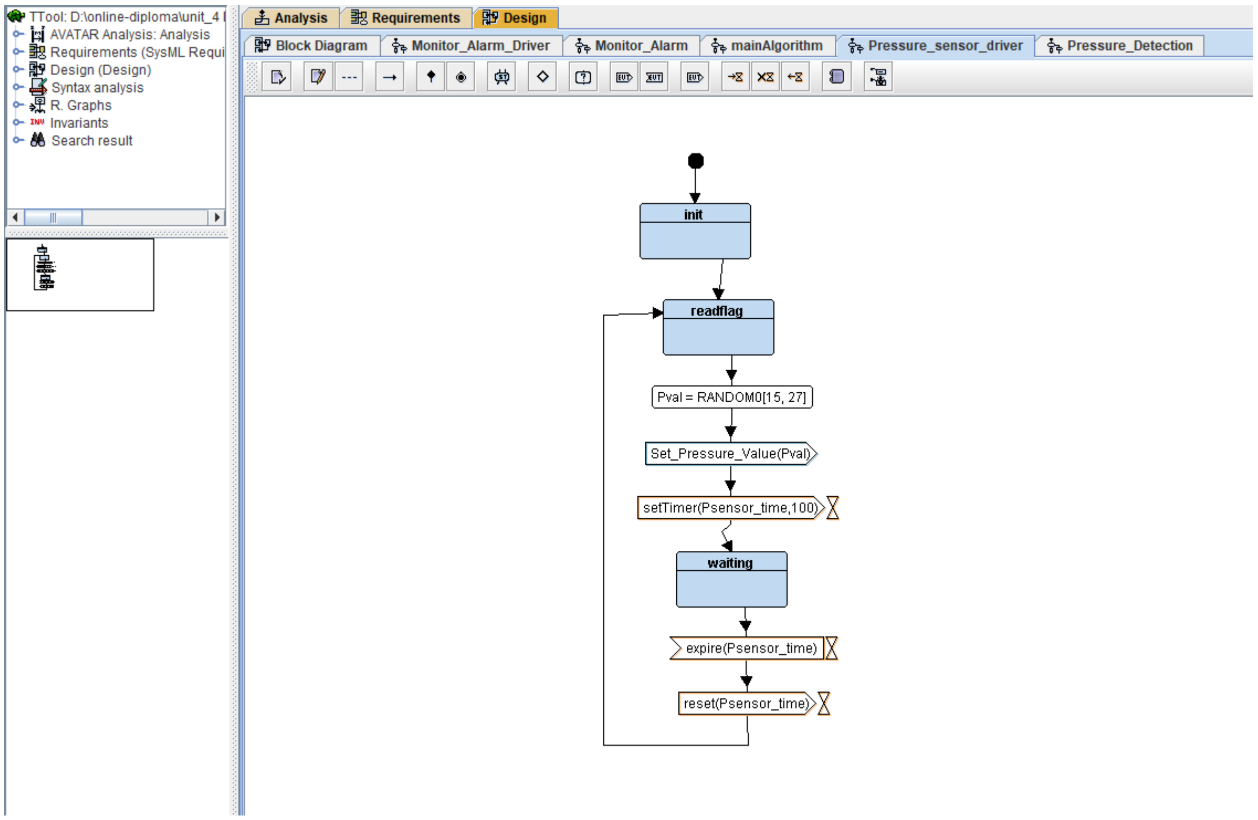
System Design



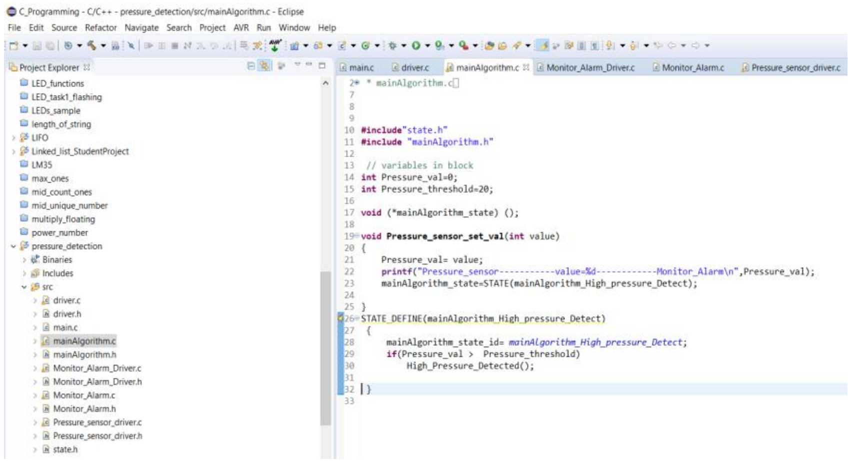



TTool: D:\online-diploma\unit_4
AVATAR Analysis: Analysis
Requirements (SysML Requi
Design (Design)
Syntax analysis
R. Graphs
Invariants
Search result
Analysis
Requirements
Design
Block Diagram
Monitor_Alarm_Driver
Monitor_Alarm
mainAlgorithm
Pressure_sensor_driver
Pressure_Detection
Alarm_OFF
Stop_Alarm()
High_Pressure_Detected()
AlarmON
Start_Alarm()
setTimer(Alarm_Timer,60)
waiting
expire(Alarm_Timer)
reset(Alarm_Timer)

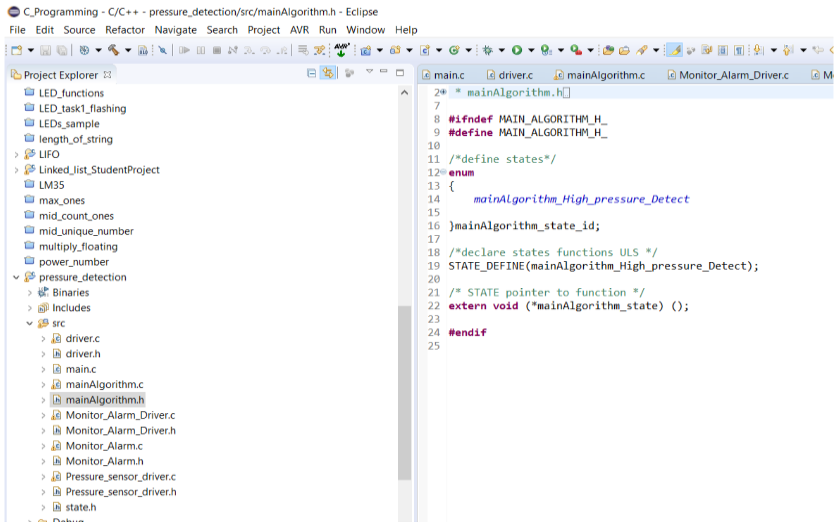

TTool: D:\online-diploma\unit_4
AVATAR Analysis: Analysis
Requirements (SysML Requi
Design (Design)
Syntax analysis
R. Graphs
Invariants
Search result
Analysis
Requirements
Design
Block Diagram
Monitor_Alarm_Driver
Monitor_Alarm
mainAlgorithm
Pressure_sensor_driver
Pressure_Detection
Set_Pressure_Value(Pval)
High_Pressure_Detect
[Pval > Threshold]
[Pval <= Threshold]
High_Pressure_Detected()
Set_Pressure_Value(Pval)
Set_Pressure_Value(Pval)


TTool: D:\online-diploma\unit_4
AVATAR Analysis: Analysis
Requirements (SysML Requi
Design (Design)
Syntax analysis
R. Graphs
Invariants
Search result
Analysis
Requirements
Design
Block Diagram
Monitor_Alarm_Driver
Monitor_Alarm
mainAlgorithm
Pressure_sensor_driver
Pressure_Detection
init
readflag
Pval = RANDOM0[15, 27]
Set_Pressure_Value(Pval)
setTimer(Psensor_time,100)
waiting
expire(Psensor_time)
reset(Psensor_time)

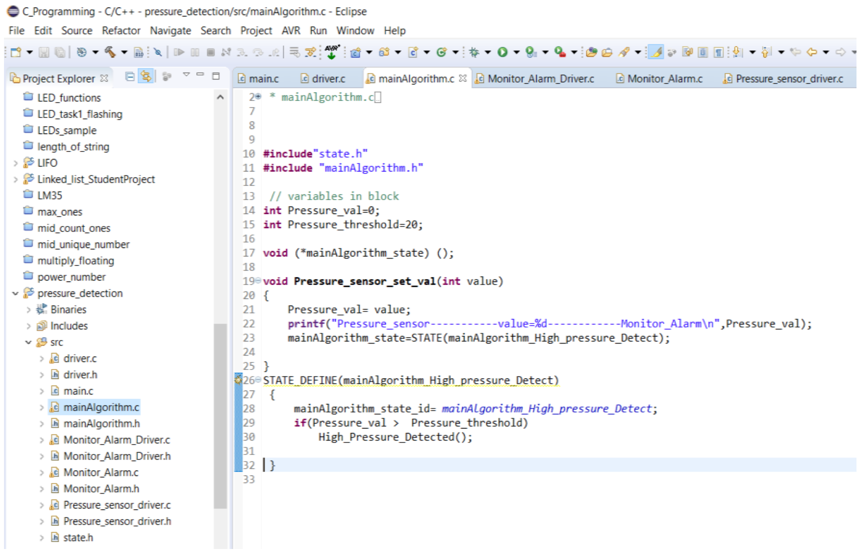
Source Code



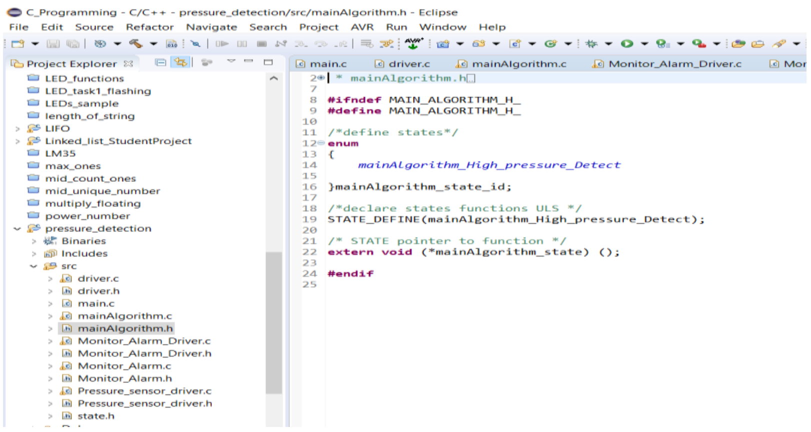
```c
 * mainAlgorithm.c



#include"state.h"
#include "mainAlgorithm.h"

 // variables in block
int Pressure_val=0;
int Pressure_threshold=20;

void (*mainAlgorithm_state) ();

void Pressure_sensor_set_val(int value)
{
    Pressure_val= value;
    printf("Pressure_sensor----------value=%d-----------Monitor_Alarm\n",Pressure_val);
    mainAlgorithm_state=STATE(mainAlgorithm_High_pressure_Detect);

}
STATE_DEFINE(mainAlgorithm_High_pressure_Detect)
 {
     mainAlgorithm_state_id= mainAlgorithm_High_pressure_Detect;
     if(Pressure_val >  Pressure_threshold)
         High_Pressure_Detected();

 }
```



```c
 * mainAlgorithm.h

#ifndef MAIN_ALGORITHM_H_
#define MAIN_ALGORITHM_H_

/*define states*/
enum
{
    mainAlgorithm_High_pressure_Detect

}mainAlgorithm_state_id;

/*declare states functions ULS */
STATE_DEFINE(mainAlgorithm_High_pressure_Detect);

/* STATE pointer to function */
extern void (*mainAlgorithm_state) ();

#endif
```

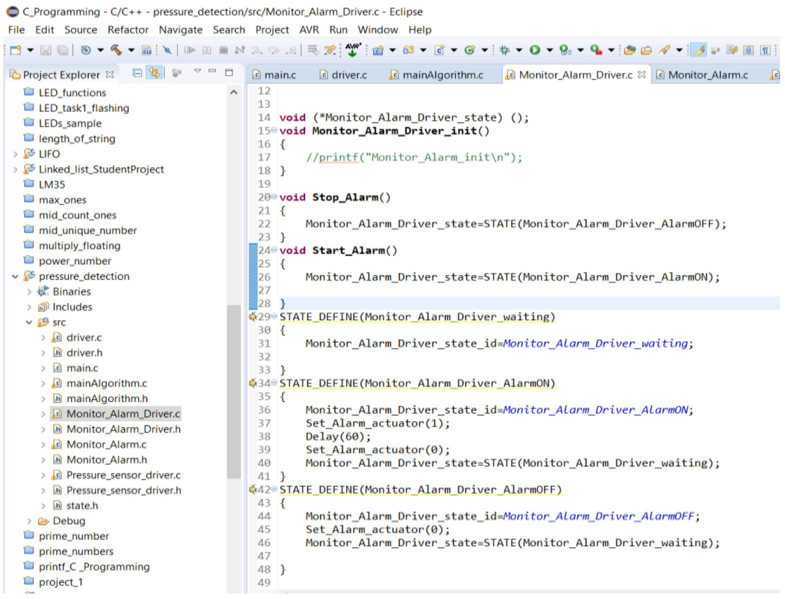


```c
 * mainAlgorithm.c



#include"state.h"
#include "mainAlgorithm.h"

 // variables in block
int Pressure_val=0;
int Pressure_threshold=20;

void (*mainAlgorithm_state) ();

void Pressure_sensor_set_val(int value)
{
    Pressure_val= value;
    printf("Pressure_sensor----------value=%d-----------Monitor_Alarm\n",Pressure_val);
    mainAlgorithm_state=STATE(mainAlgorithm_High_pressure_Detect);

}
STATE_DEFINE(mainAlgorithm_High_pressure_Detect)
 {
     mainAlgorithm_state_id= mainAlgorithm_High_pressure_Detect;
     if(Pressure_val >  Pressure_threshold)
         High_Pressure_Detected();

 }
```



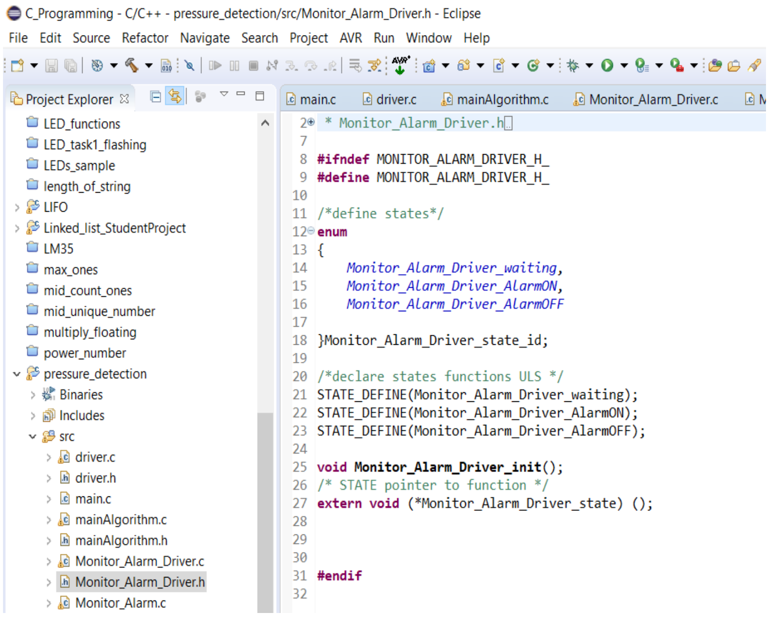
C_Programming - C/C++ - pressure_detection/src/mainAlgorithm.h - Eclipse

```c
 * mainAlgorithm.h

#ifndef MAIN_ALGORITHM_H_
#define MAIN_ALGORITHM_H_

/*define states*/
enum
{
	mainAlgorithm_High_pressure_Detect

}mainAlgorithm_state_id;

/*declare states functions ULS */
STATE_DEFINE(mainAlgorithm_High_pressure_Detect);

/* STATE pointer to function */
extern void (*mainAlgorithm_state) ();

#endif
```



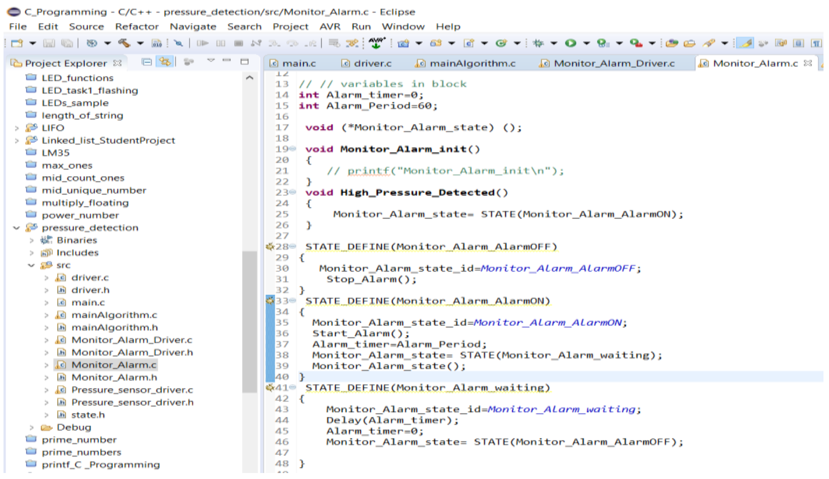
C_Programming - C/C++ - pressure_detection/src/Monitor_Alarm_Driver.c - Eclipse

```c


void (*Monitor_Alarm_Driver_state) ();
void Monitor_Alarm_Driver_init()
{
	//printf("Monitor_Alarm_init\n");
}

void Stop_Alarm()
{
	Monitor_Alarm_Driver_state=STATE(Monitor_Alarm_Driver_AlarmOFF);
}
void Start_Alarm()
{
	Monitor_Alarm_Driver_state=STATE(Monitor_Alarm_Driver_AlarmON);

}
STATE_DEFINE(Monitor_Alarm_Driver_waiting)
{
	Monitor_Alarm_Driver_state_id=Monitor_Alarm_Driver_waiting;

}
STATE_DEFINE(Monitor_Alarm_Driver_AlarmON)
{
	Monitor_Alarm_Driver_state_id=Monitor_Alarm_Driver_AlarmON;
	Set_Alarm_actuator(1);
	Delay(60);
	Set_Alarm_actuator(0);
	Monitor_Alarm_Driver_state=STATE(Monitor_Alarm_Driver_waiting);
}
STATE_DEFINE(Monitor_Alarm_Driver_AlarmOFF)
{
	Monitor_Alarm_Driver_state_id=Monitor_Alarm_Driver_AlarmOFF;
	Set_Alarm_actuator(0);
	Monitor_Alarm_Driver_state=STATE(Monitor_Alarm_Driver_waiting);

}
```

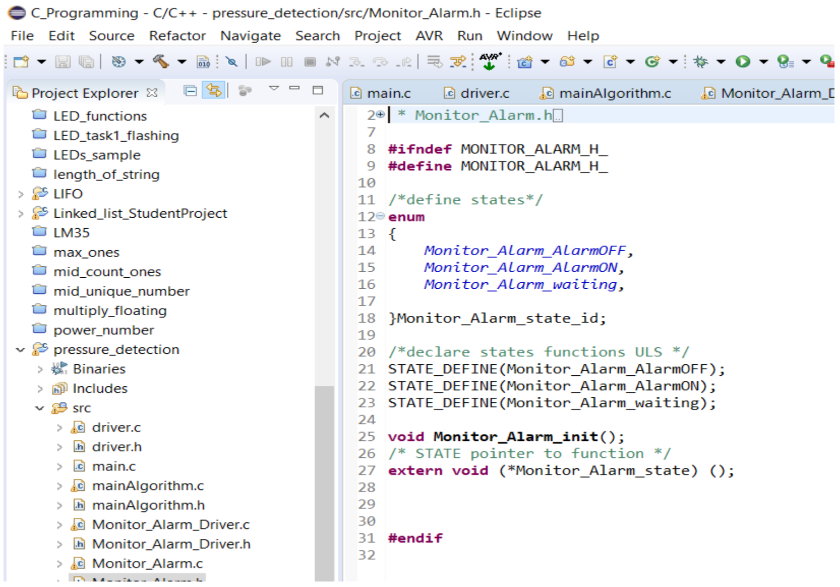


C_Programming - C/C++ - pressure_detection/src/Monitor_Alarm_Driver.h - Eclipse

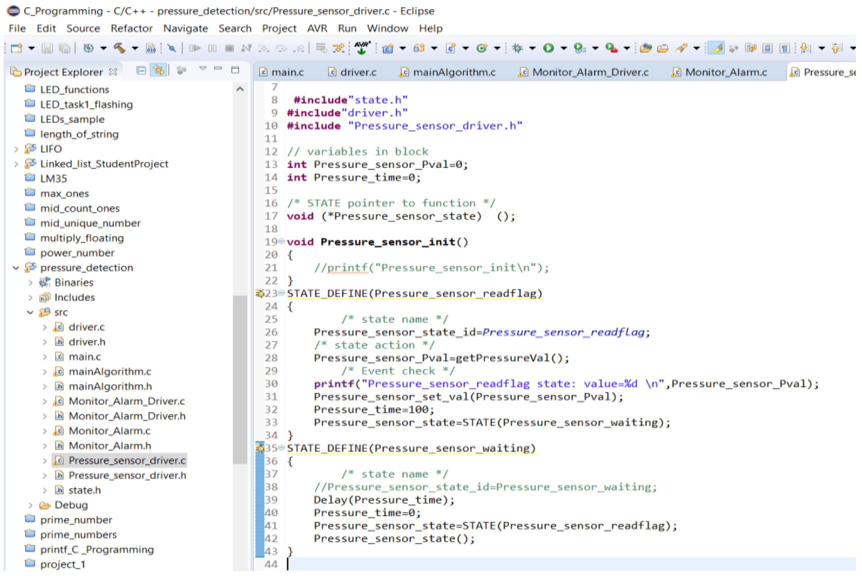
```c
 * Monitor_Alarm_Driver.h

#ifndef MONITOR_ALARM_DRIVER_H_
#define MONITOR_ALARM_DRIVER_H_

/*define states*/
enum
{
	Monitor_Alarm_Driver_waiting,
	Monitor_Alarm_Driver_AlarmON,
	Monitor_Alarm_Driver_AlarmOFF

}Monitor_Alarm_Driver_state_id;

/*declare states functions ULS */
STATE_DEFINE(Monitor_Alarm_Driver_waiting);
STATE_DEFINE(Monitor_Alarm_Driver_AlarmON);
STATE_DEFINE(Monitor_Alarm_Driver_AlarmOFF);

void Monitor_Alarm_Driver_init();
/* STATE pointer to function */
extern void (*Monitor_Alarm_Driver_state) ();



#endif
```


C_Programming - C/C++ - pressure_detection/src/Monitor_Alarm.c - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
main.c driver.c mainAlgorithm.c Monitor_Alarm_Driver.c Monitor_Alarm.c

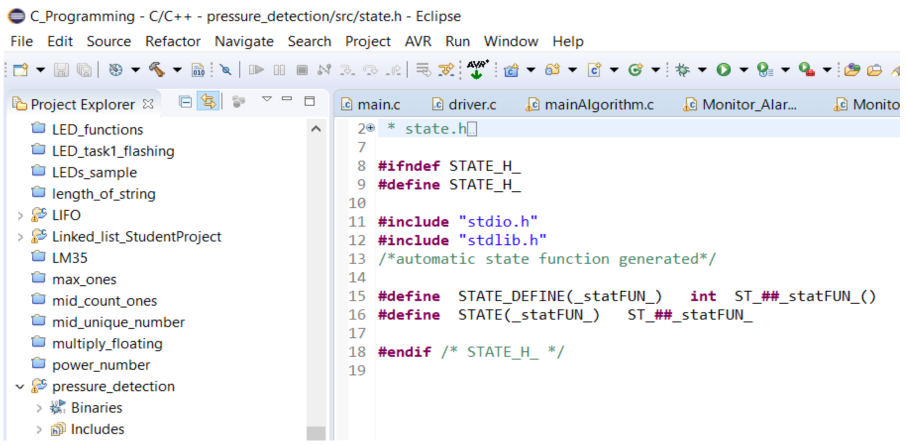
```c
// // variables in block
int Alarm_timer=0;
int Alarm_Period=60;

 void (*Monitor_Alarm_state) ();

 void Monitor_Alarm_init()
 {
    // printf("Monitor_Alarm_init\n");
 }
 void High_Pressure_Detected()
 {
     Monitor_Alarm_state= STATE(Monitor_Alarm_AlarmON);
 }

 STATE_DEFINE(Monitor_Alarm_AlarmOFF)
{
   Monitor_Alarm_state_id=Monitor_Alarm_AlarmOFF;
    Stop_Alarm();
}
 STATE_DEFINE(Monitor_Alarm_AlarmON)
{
  Monitor_Alarm_state_id=Monitor_Alarm_AlarmON;
  Start_Alarm();
  Alarm_timer=Alarm_Period;
  Monitor_Alarm_state= STATE(Monitor_Alarm_waiting);
  Monitor_Alarm_state();
}
 STATE_DEFINE(Monitor_Alarm_waiting)
{
    Monitor_Alarm_state_id=Monitor_Alarm_waiting;
    Delay(Alarm_timer);
    Alarm_timer=0;
    Monitor_Alarm_state= STATE(Monitor_Alarm_AlarmOFF);

}
```


C_Programming - C/C++ - pressure_detection/src/Monitor_Alarm.h - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
main.c driver.c mainAlgorithm.c Monitor_Alarm_D

```c
* Monitor_Alarm.h

#ifndef MONITOR_ALARM_H_
#define MONITOR_ALARM_H_

/*define states*/
enum
{
    Monitor_Alarm_AlarmOFF,
    Monitor_Alarm_AlarmON,
    Monitor_Alarm_waiting,

}Monitor_Alarm_state_id;

/*declare states functions ULS */
STATE_DEFINE(Monitor_Alarm_AlarmOFF);
STATE_DEFINE(Monitor_Alarm_AlarmON);
STATE_DEFINE(Monitor_Alarm_waiting);

void Monitor_Alarm_init();
/* STATE pointer to function */
extern void (*Monitor_Alarm_state) ();



#endif
```

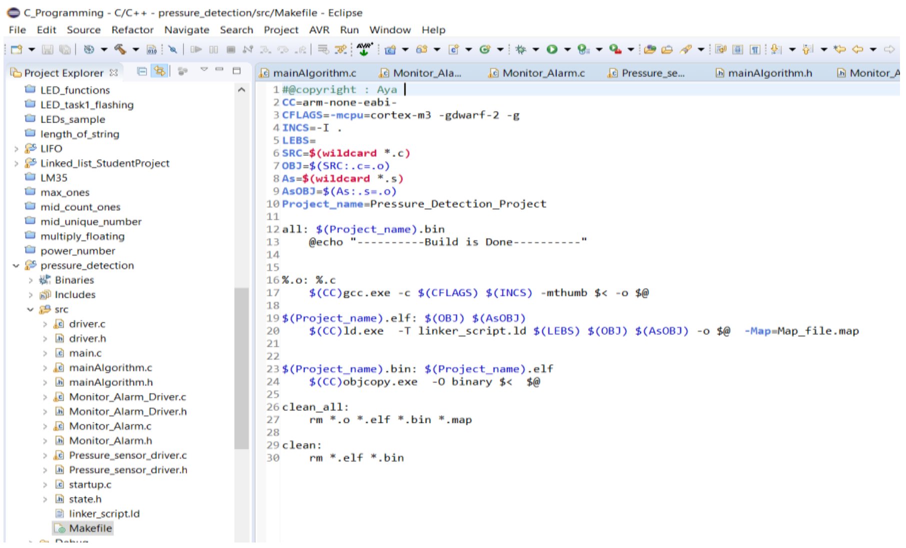

C_Programming - C/C++ - pressure_detection/src/Pressure_sensor_driver.c - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help
Project Explorer
main.c driver.c mainAlgorithm.c Monitor_Alarm_Driver.c Monitor_Alarm.c Pressure_se

```c

 #include"state.h"
#include"driver.h"
#include "Pressure_sensor_driver.h"

// variables in block
int Pressure_sensor_Pval=0;
int Pressure_time=0;

/* STATE pointer to function */
void (*Pressure_sensor_state)  ();

void Pressure_sensor_init()
{
    //printf("Pressure_sensor_init\n");
}
STATE_DEFINE(Pressure_sensor_readflag)
{
        /* state name */
    Pressure_sensor_state_id=Pressure_sensor_readflag;
    /* state action */
    Pressure_sensor_Pval=getPressureVal();
        /* Event check */
    printf("Pressure_sensor_readflag state: value=%d \n",Pressure_sensor_Pval);
    Pressure_sensor_set_val(Pressure_sensor_Pval);
    Pressure_time=100;
    Pressure_sensor_state=STATE(Pressure_sensor_waiting);
}
STATE_DEFINE(Pressure_sensor_waiting)
{
        /* state name */
    //Pressure_sensor_state_id=Pressure_sensor_waiting;
    Delay(Pressure_time);
    Pressure_time=0;
    Pressure_sensor_state=STATE(Pressure_sensor_readflag);
    Pressure_sensor_state();
}

```



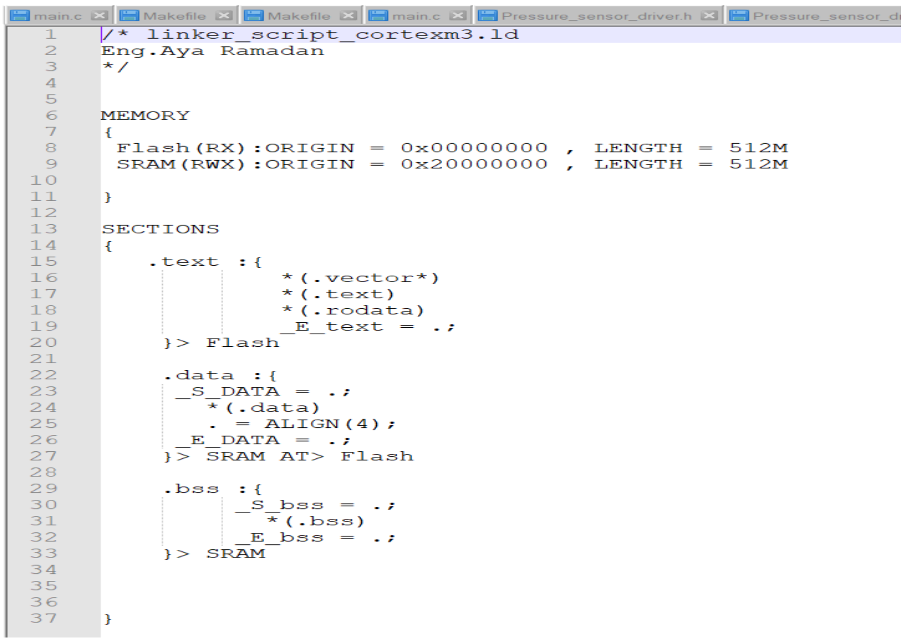
C_Programming - C/C++ - pressure_detection/src/state.h - Eclipse

```c
 * state.h

#ifndef STATE_H_
#define STATE_H_

#include "stdio.h"
#include "stdlib.h"
/*automatic state function generated*/

#define  STATE_DEFINE(_statFUN_)   int  ST_##_statFUN_()
#define  STATE(_statFUN_)    ST_##_statFUN_

#endif /* STATE_H_ */
```

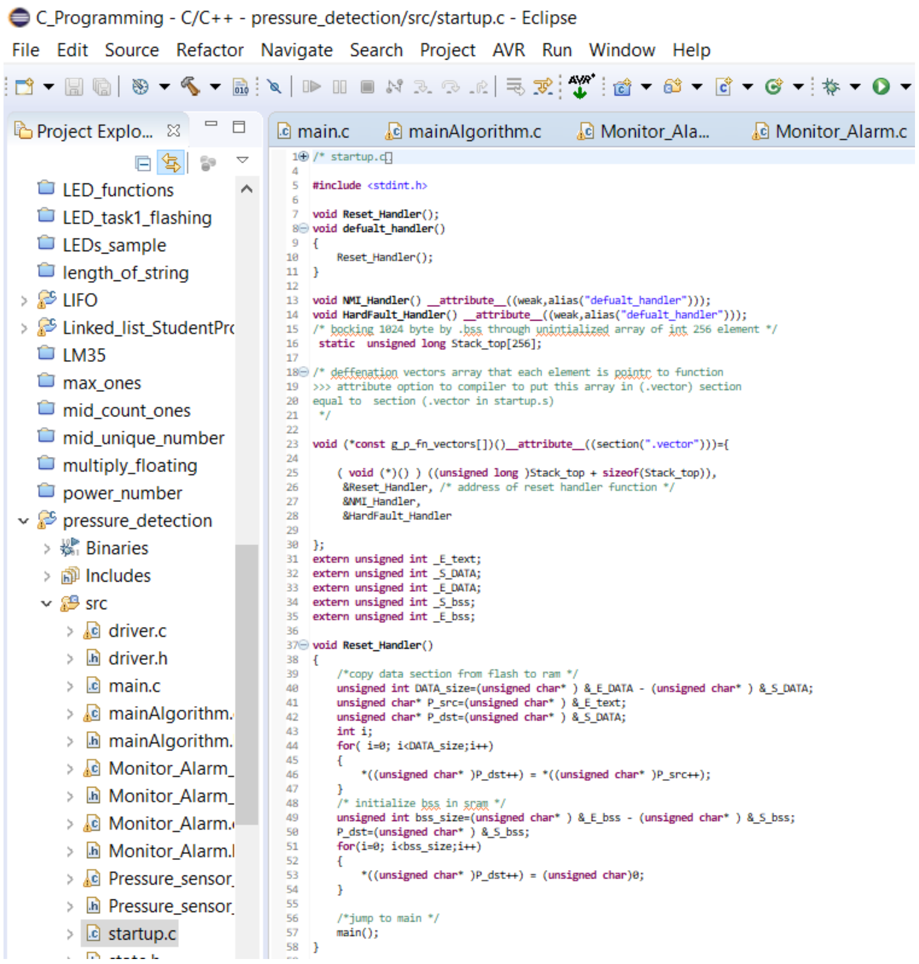


C_Programming - C/C++ - pressure_detection/src/Makefile - Eclipse

```makefile
#@copyright : Aya
CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -gdwarf-2 -g
INCS=-I .
LEBS=
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
As=$(wildcard *.s)
AsOBJ=$(As:.s=.o)
Project_name=Pressure_Detection_Project

all: $(Project_name).bin
	@echo "----------Build is Done----------"


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

$(Project_name).elf: $(OBJ) $(AsOBJ)
	$(CC)ld.exe  -T linker_script.ld $(LEBS) $(OBJ) $(AsOBJ) -o $@  -Map=Map_file.map


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

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

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

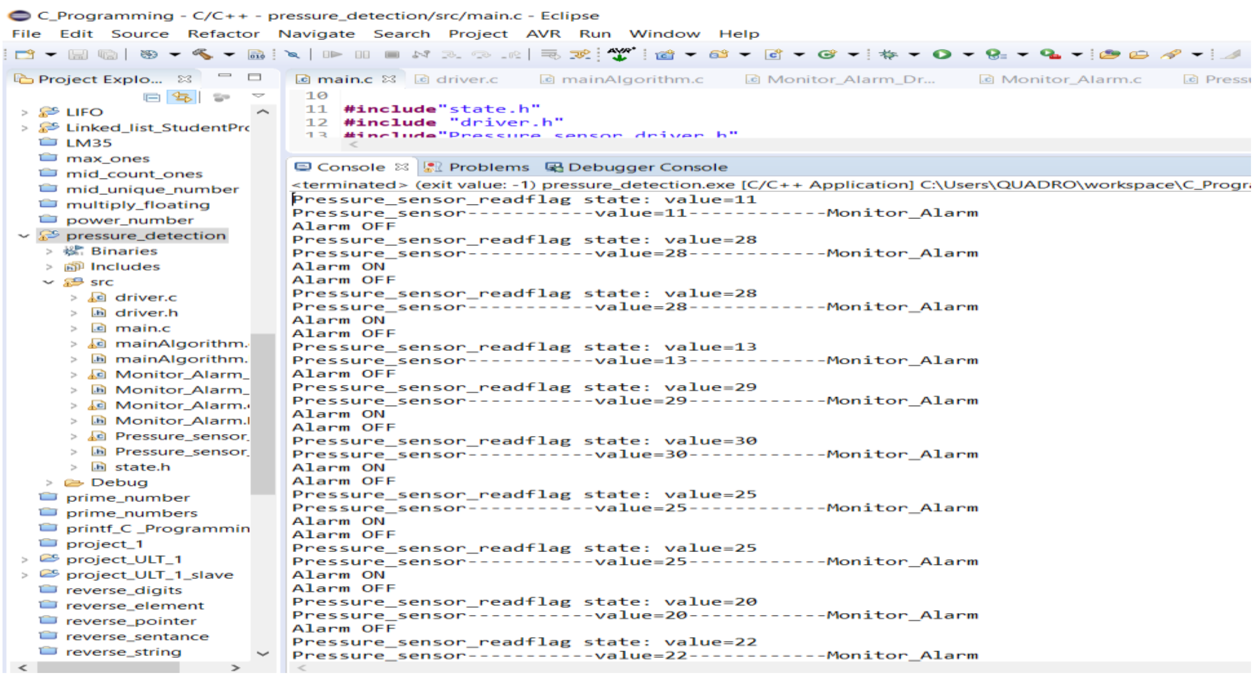


```
/* linker_script_cortexm3.ld
Eng.Aya Ramadan
*/


MEMORY
{
 Flash(RX):ORIGIN = 0x00000000 , LENGTH = 512M
 SRAM(RWX):ORIGIN = 0x20000000 , LENGTH = 512M

}

SECTIONS
{
   .text  :{
                 *(.vector*)
                 *(.text)
                 *(.rodata)
                  _E_text = .;
    }> Flash

    .data  :{
     _S_DATA = .;
       *(.data)
       .  = ALIGN(4);
     _E_DATA = .;
    }> SRAM AT> Flash

    .bss  :{
         _S_bss = .;
           *(.bss)
         _E_bss = .;
    }> SRAM



}
```



```c
/* startup.c

#include <stdint.h>

void Reset_Handler();
void defualt_handler()
{
    Reset_Handler();
}

void NMI_Handler() __attribute__((weak,alias("defualt_handler")));
void HardFault_Handler() __attribute__((weak,alias("defualt_handler")));
/* bocking 1024 byte by .bss through unintialized array of int 256 element */
 static  unsigned long Stack_top[256];

/* deffenation vectors array that each element is pointr to function
>>> attribute option to compiler to put this array in (.vector) section
equal to  section (.vector in startup.s)
 */

void (*const g_p_fn_vectors[])()__attribute__((section(".vector")))={

    ( void (*)() ) ((unsigned long )Stack_top + sizeof(Stack_top)),
     &Reset_Handler, /* address of reset handler function */
     &NMI_Handler,
     &HardFault_Handler

};
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;

void Reset_Handler()
{
    /*copy data section from flash to ram */
    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;
    int i;
    for( i=0; i<DATA_size;i++)
    {
        *((unsigned char* )P_dst++) = *((unsigned char* )P_src++);
    }
    /* initialize bss in sram */
    unsigned int bss_size=(unsigned char* ) &_E_bss - (unsigned char* ) &_S_bss;
    P_dst=(unsigned char* ) &_S_bss;
    for(i=0; i<bss_size;i++)
    {
        *((unsigned char* )P_dst++) = (unsigned char)0;
    }

    /*jump to main */
    main();
}
```

Testing



```c
#include"state.h"
#include "driver.h"
#include"Pressure_sensor_driver.h"
```

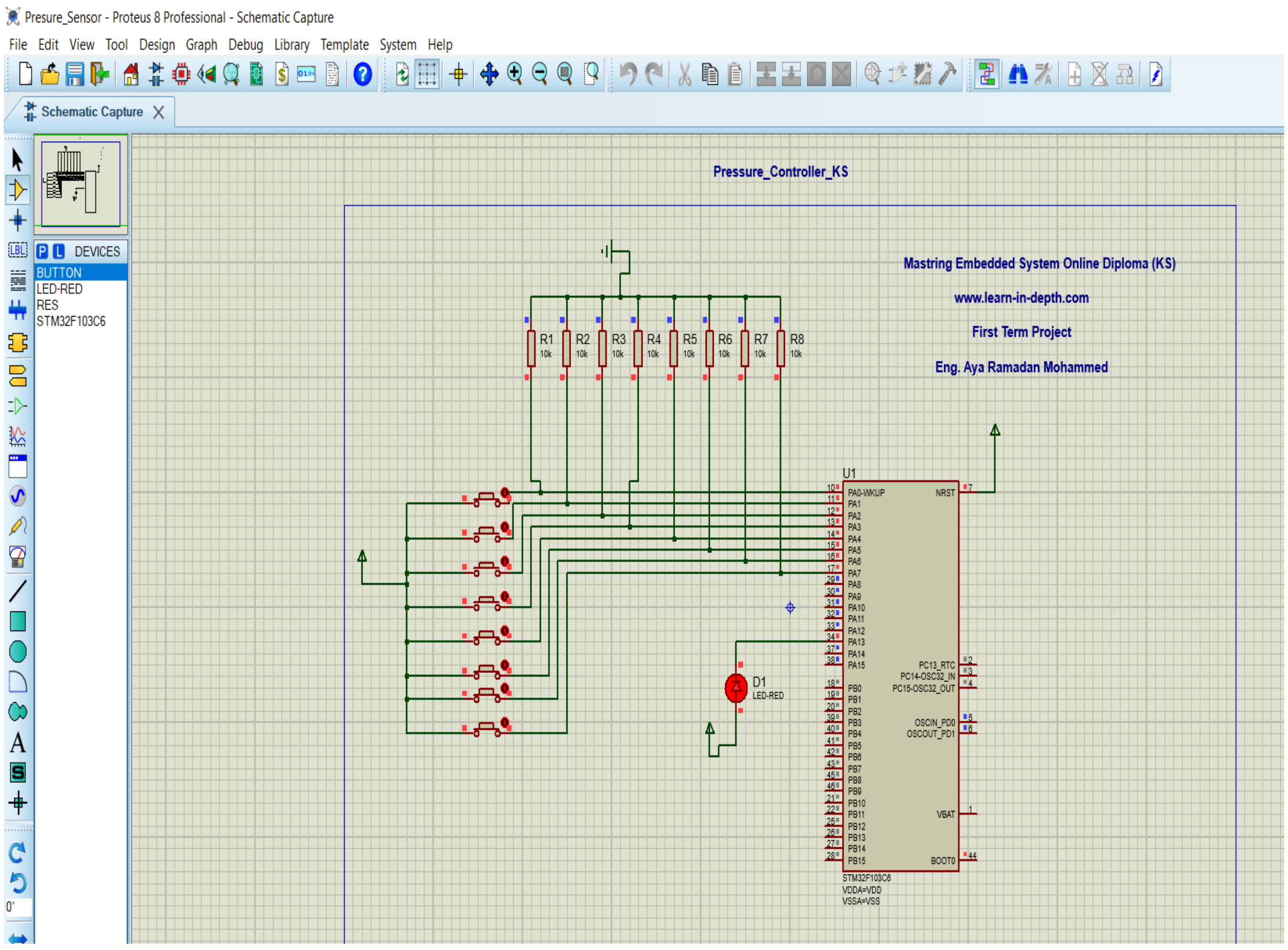
```
<terminated> (exit value: -1) pressure_detection.exe [C/C++ Application] C:\Users\QUADRO\workspace\C_Progr
Pressure_sensor_readflag state:  value=11
Pressure_sensor-----------value=11------------Monitor_Alarm
Alarm OFF
Pressure_sensor_readflag state:  value=28
Pressure_sensor-----------value=28------------Monitor_Alarm
Alarm ON
Alarm OFF
Pressure_sensor_readflag state:  value=28
Pressure_sensor-----------value=28------------Monitor_Alarm
Alarm ON
Alarm OFF
Pressure_sensor_readflag state:  value=13
Pressure_sensor-----------value=13------------Monitor_Alarm
Alarm OFF
Pressure_sensor_readflag state:  value=29
Pressure_sensor-----------value=29------------Monitor_Alarm
Alarm ON
Alarm OFF
Pressure_sensor_readflag state:  value=30
Pressure_sensor-----------value=30------------Monitor_Alarm
Alarm ON
Alarm OFF
Pressure_sensor_readflag state:  value=25
Pressure_sensor-----------value=25------------Monitor_Alarm
Alarm ON
Alarm OFF
Pressure_sensor_readflag state:  value=25
Pressure_sensor-----------value=25------------Monitor_Alarm
Alarm ON
Alarm OFF
Pressure_sensor_readflag state:  value=20
Pressure_sensor-----------value=20------------Monitor_Alarm
Alarm OFF
Pressure_sensor_readflag state:  value=22
Pressure_sensor-----------value=22------------Monitor_Alarm
```

Case 1	Case 2

`Pressure_sensor_readflag state: value=11`
`Pressure_sensor-----------value=11------------Monitor_Alarm`
`Alarm OFF`
	`Pressure_sensor_readflag state: value=28`
	`Pressure_sensor-----------value=28------------Monitor_Alarm`
	`Alarm ON`
	`Alarm OFF`

In this case the pressure =11 (Less than threshold) … The pressure sensor sent this value to main algorithm and then sent to monitor alarm and because it is less than 20 bar the monitor alarm to alarm actuator to stop .	In this case the pressure =28 (Bigger than threshold)… The pressure sensor sent this value to main algorithm and then sent to monitor and because it is bigger than 20 bar the monitor alarm sent to alarm actuator to Start and wait 60s then stop.

Simulation


Presure_Sensor - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
DEVICES
BUTTON
LED-RED
RES
STM32F103C6
Pressure_Controller_KS
Mastring Embedded System Online Diploma (KS)
www.learn-in-depth.com
First Term Project
Eng. Aya Ramadan Mohammed
R1 R2 R3 R4 R5 R6 R7 R8
10k
U1
D1
LED-RED
STM32F103C6
VDDA=VDD
VSSA=VSS