

Mastering Embedded System Online Diploma

[www.learn-in-depth.com](http://www.learn-in-depth.com)

First Term (Final Project 1)

Eng. Ahmed Sameh

My Profile: <https://www.learn-in-depth.com/online-diploma/ahmeds.elshahid%40gmail.com>

## Contents

Table of Figures:.....	3
Requirements Diagram: .....	4
System Analysis:.....	5
Use Case Diagram .....	5
Activity Diagram .....	6
Sequence Diagram .....	7
System Design:.....	8
Block Diagram .....	8
Pressure Driver State Diagram .....	9
Main Alg. State Diagram .....	10
Alarm Monitor State Diagram.....	11
Alarm Driver State Diagram .....	12
Codes:.....	13
.c and .h Files.....	13
Makefile .....	24
Linker file.....	25
Startup file.....	26
Map_file.map.....	27
Symbol Table.....	28
Simulation Results:.....	29

## Table of Figures:

Figure 1: Requirements Diagram .....	4
Figure 2: Use Case Diagram .....	5
Figure 3: Activity Diagram.....	6
Figure 4: Sequence Diagram .....	7
Figure 5: Block Diagram .....	8
Figure 6: Pressure Driver State Diagram.....	9
Figure 7: Main Alg. State Diagram .....	10
Figure 8: Alarm Monitor State Diagram.....	11
Figure 9: Alarm Driver State Diagram .....	12
Figure 10: main.c.....	13
Figure 11: driver.c .....	14
Figure 12: driver.h.....	15
Figure 13: alarmDriver.c.....	16
Figure 14: alarmDriver.c.....	17
Figure 15: alarmMonitor.c .....	18
Figure 16: alarmMonitor.h.....	19
Figure 17: mainAlg.c.....	19
Figure 18: mainAlg.h .....	20
Figure 19: pressureDriver.c.....	21
Figure 20: pressureDriver.h .....	22
Figure 21: stateMachine.h .....	22
Figure 22: platformTypes.h.....	23
Figure 23: Makefile .....	24
Figure 24: linker_script.ld .....	25
Figure 25: startup.c .....	26
Figure 26: Map_file.map.....	27
Figure 27: symbol Table .....	28
Figure 28: simulation results when pressure is less than 20 .....	29
Figure 29: simulation results when pressure is larger than or equal to 20 .....	30

## Requirements Diagram:

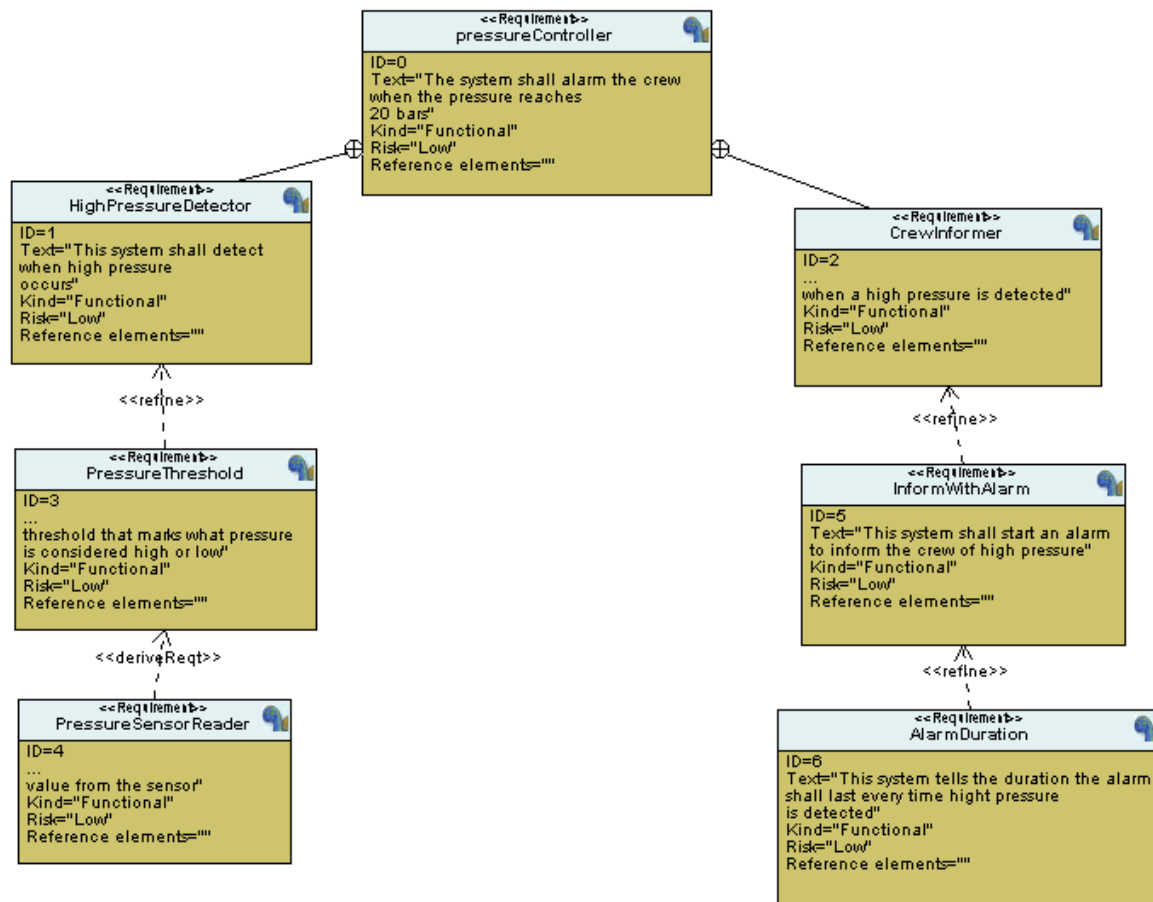
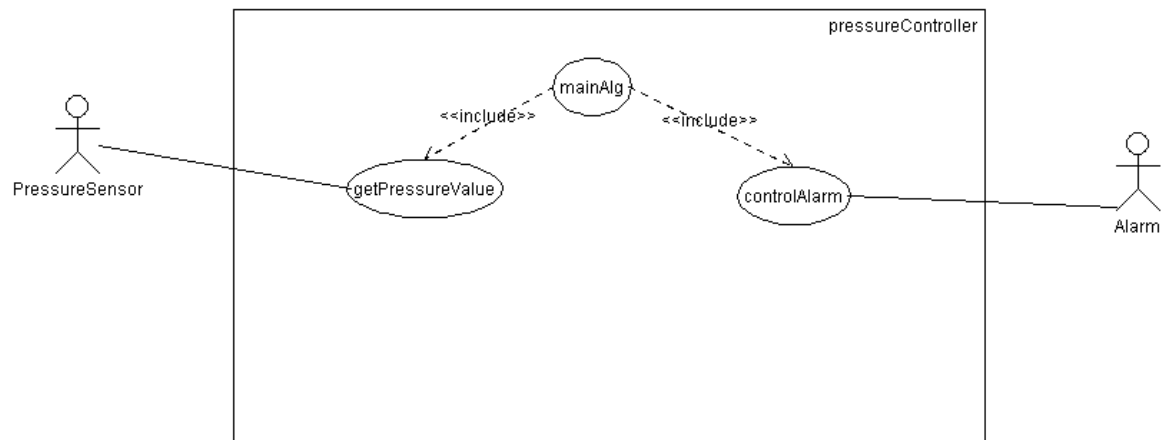


Figure 1: Requirements Diagram

System Analysis:  
Use Case Diagram



*Figure 2: Use Case Diagram*

---

## Activity Diagram

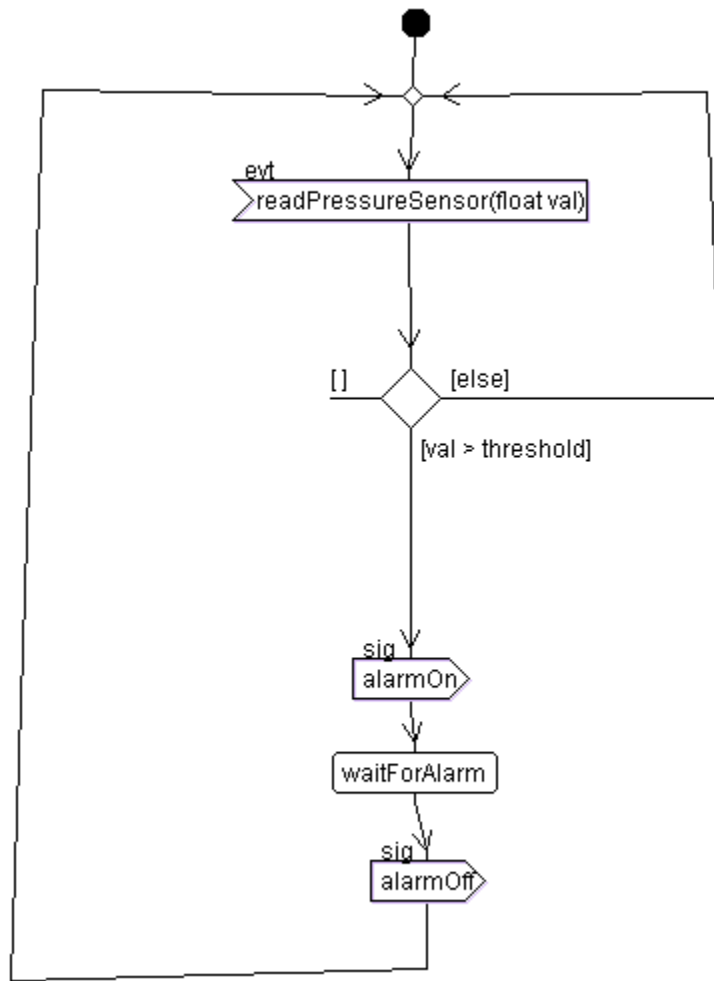


Figure 3: Activity Diagram

## Sequence Diagram

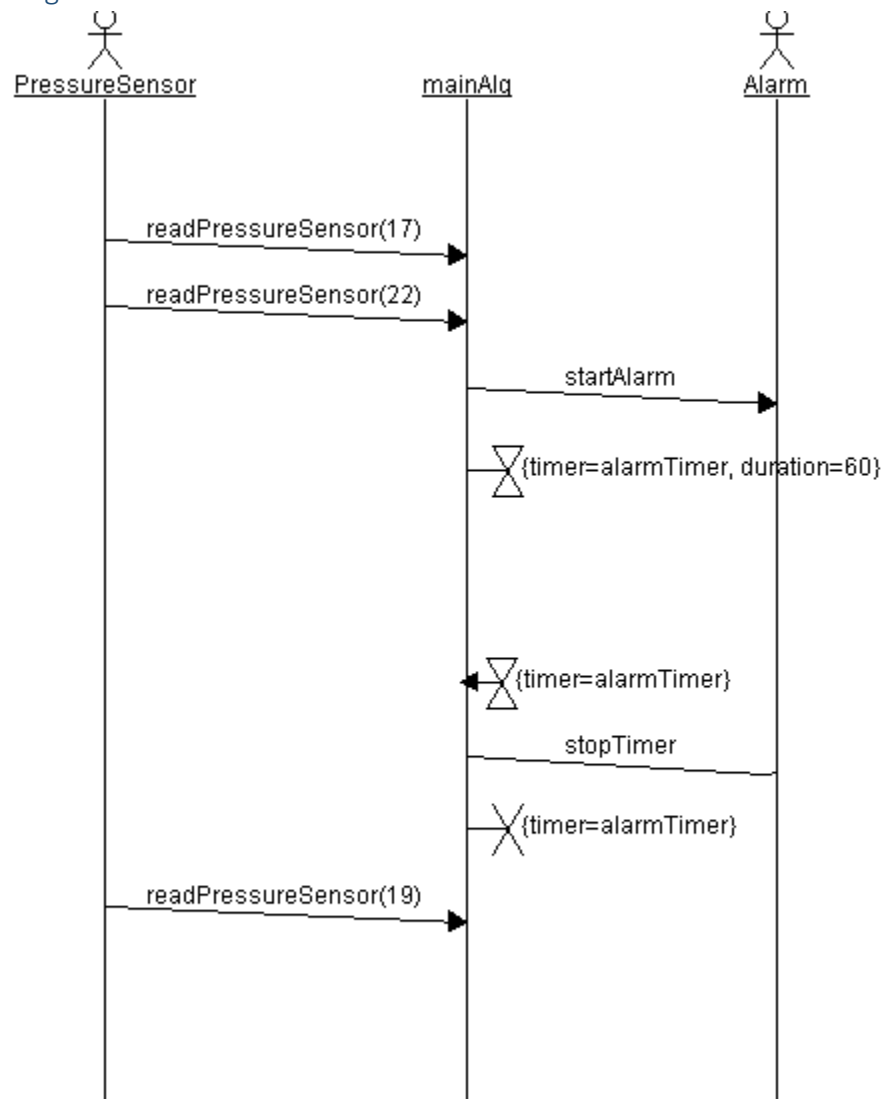


Figure 4: Sequence Diagram

## System Design:

### Block Diagram

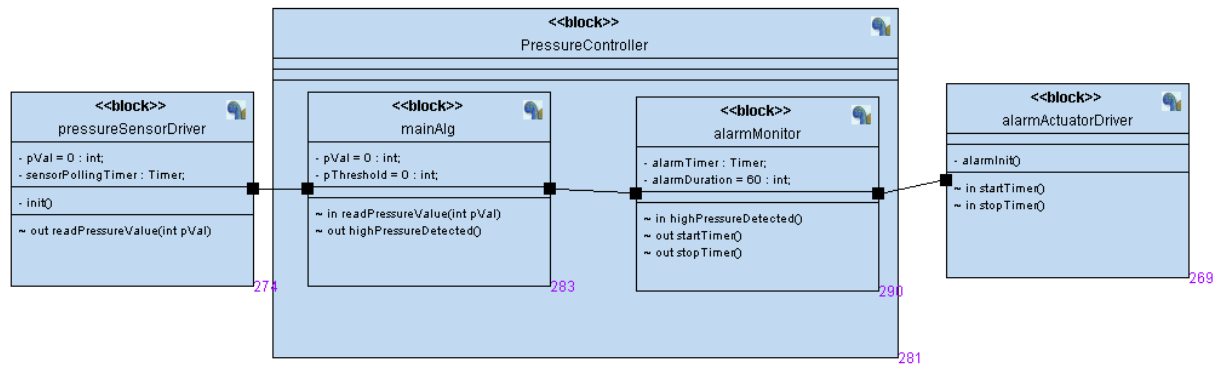


Figure 5: Block Diagram



## Pressure Driver State Diagram

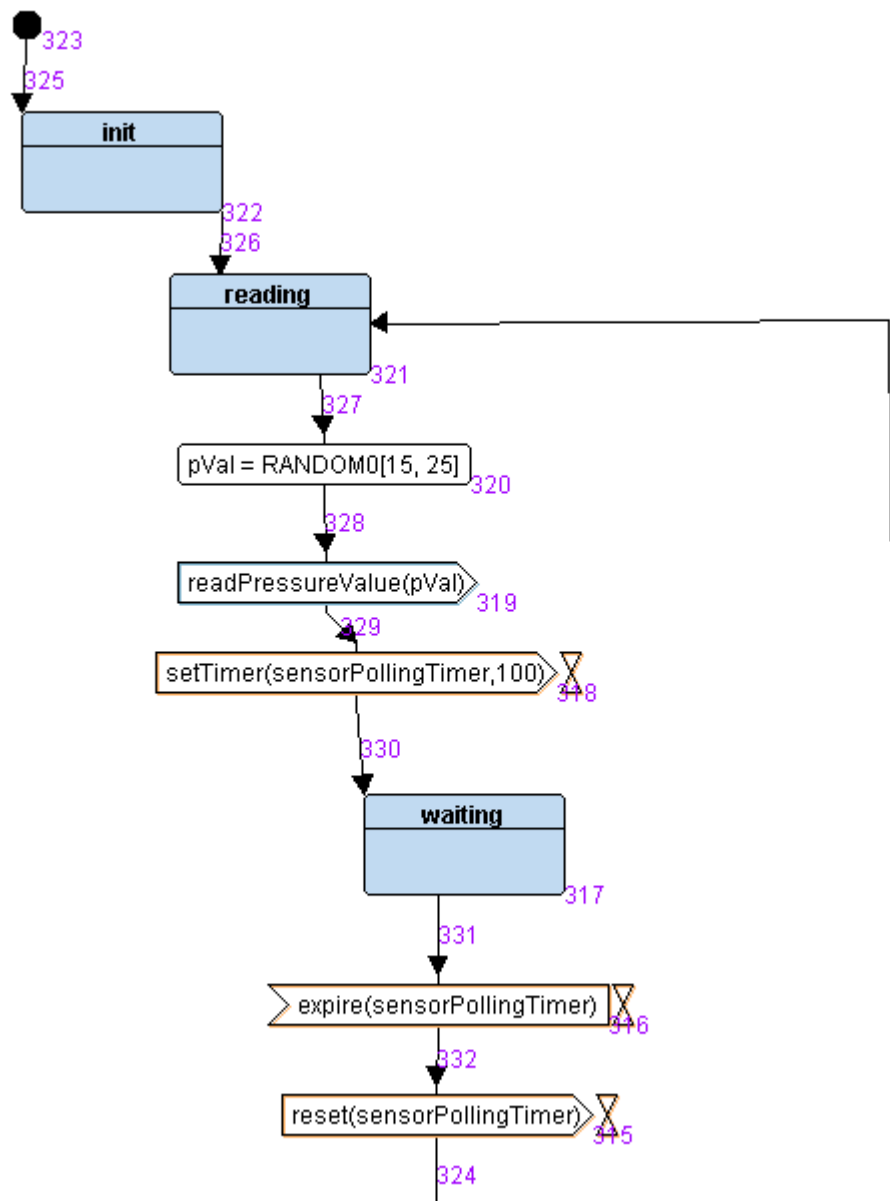


Figure 6: Pressure Driver State Diagram

## Main Alg. State Diagram

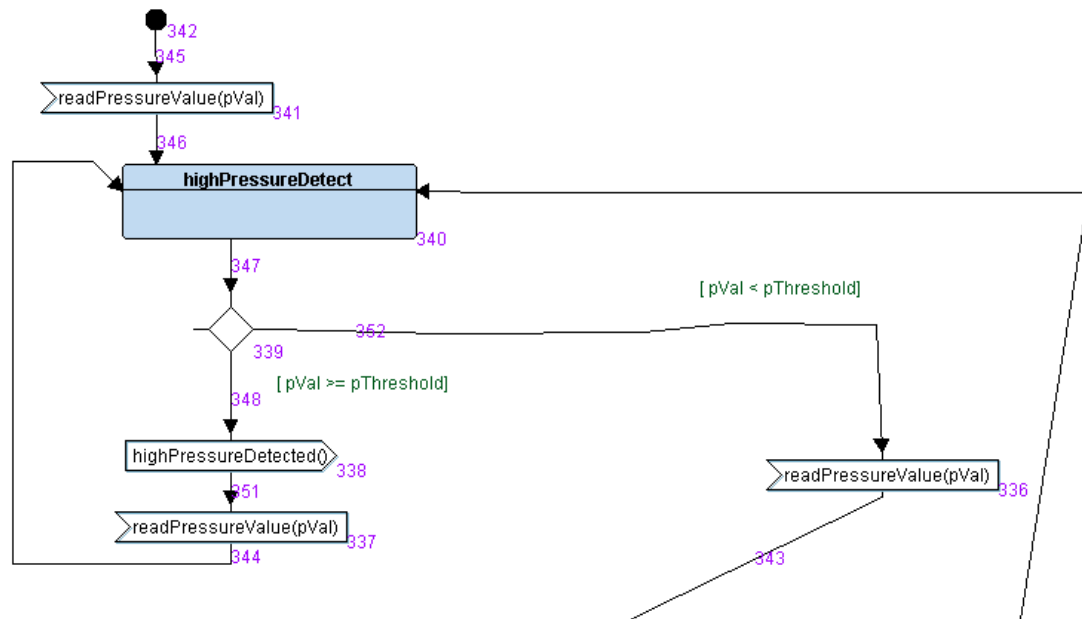


Figure 7: Main Alg. State Diagram

## Alarm Monitor State Diagram

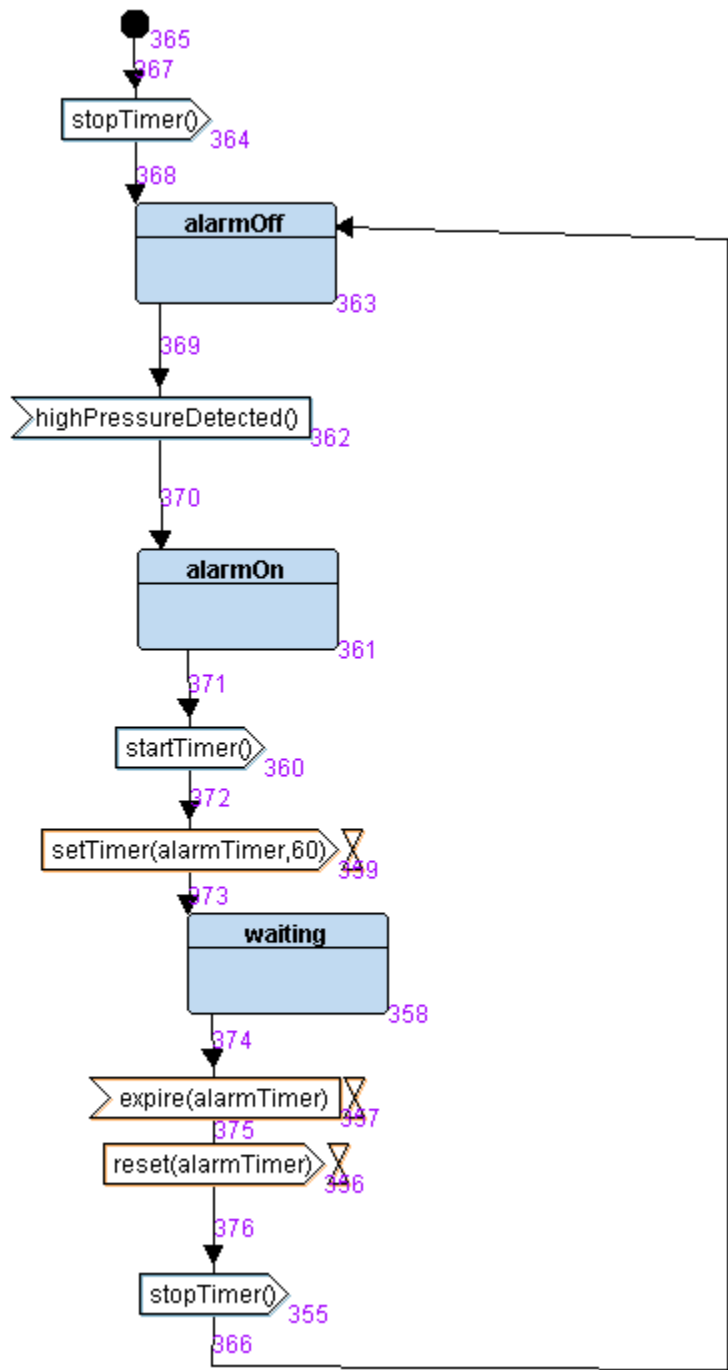


Figure 8: Alarm Monitor State Diagram

## Alarm Driver State Diagram

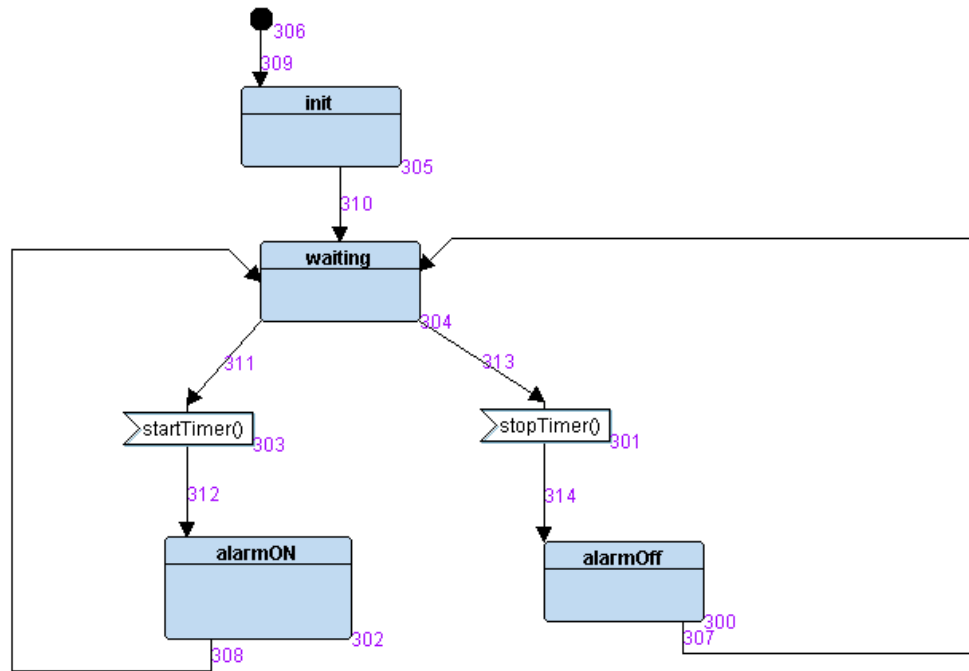


Figure 9: Alarm Driver State Diagram

## Codes:

.c and .h Files

```
4  #include "alarmMonitor.h"
5  #include "mainAlg.h"
6  #include "alarmDriver.h"
7
8
9  void setup()
10 {
11     GPIO_INITIALIZATION();
12
13     pressure_init();
14     alarmDriver_init();
15
16
17     PRESSURE_state = STATE(PRESSURE_reading);
18     MAIN_ALG_state = STATE(MAIN_ALG_highPressureDetect);
19     ALARM_MONITOR_state = STATE(ALARM_MONITOR_alarmOFF);
20     ALARM_DRIVER_state = STATE(ALARM_DRIVER_waiting);
21
22 }
23
24 int main ()
25 {
26     setup();
27
28     while (1)
29     {
30         //Implement your Design
31         PRESSURE_state();
32         MAIN_ALG_state();
33         ALARM_MONITOR_state();
34         ALARM_DRIVER_state();
35     }
36
37 }
38
```

Figure 10: main.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 &= 0xFF0FFFFFF;
    GPIOA_CRL |= 0x00000000;
    GPIOA_CRH &= 0xFF0FFFFFF;
    GPIOA_CRH |= 0x22222222;
}

```

Figure 11: driver.c

```

1  #include <stdint.h>
2  #include <stdio.h>
3
4  #define SET_BIT(ADDRESS,BIT)  ADDRESS |= (1<<BIT)
5  #define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
6  #define TOGGLE_BIT(ADDRESS,BIT) ADDRESS ^= (1<<BIT)
7  #define READ_BIT(ADDRESS,BIT) ((ADDRESS) & (1<<(BIT)))
8
9
10 #define GPIO_PORTA 0x40010800
11 #define BASE_RCC   0x40021000
12
13 #define APB2ENR    *(volatile uint32_t *) (BASE_RCC + 0x18)
14
15 #define GPIOA_CRL  *(volatile uint32_t *) (GPIO_PORTA + 0x00)
16 #define GPIOA_CRH  *(volatile uint32_t *) (GPIO_PORTA + 0x04)
17 #define GPIOA_IDR  *(volatile uint32_t *) (GPIO_PORTA + 0x08)
18 #define GPIOA_ODR  *(volatile uint32_t *) (GPIO_PORTA + 0x0C)
19
20
21 void Delay(int nCount);
22 int getPressureVal();
23 void Set_Alarm_actuator(int i);
24 void GPIO_INITIALIZATION ();
25

```

Figure 12: driver.h

```

1  #include "alarmDriver.h"
2  #include "driver.h"
3
4  void (*ALARM_DRIVER_state) ();
5  void alarmDriver_init()
6  {
7      DPRINTF("ALARM INIT...\n");
8      Set_Alarm_actuator(1);
9  }
10 void startTimer()
11 {
12     ALARM_DRIVER_state = STATE(ALARM_DRIVER_alarmON);
13     ALARM_DRIVER_state();
14 }
15 void stopTimer()
16 {
17     ALARM_DRIVER_state = STATE(ALARM_DRIVER_alarmOFF);
18     ALARM_DRIVER_state();
19 }
20 STATE_define(ALARM_DRIVER_alarmON)
21 {
22     alarmDriver_state_id = alarmDriver_alarmON;
23
24     Set_Alarm_actuator(0);
25
26     ALARM_DRIVER_state = STATE(ALARM_DRIVER_waiting);
27 }
28 STATE_define(ALARM_DRIVER_alarmOFF)
29 {
30     alarmDriver_state_id = alarmDriver_alarmOFF;
31
32     Set_Alarm_actuator(1);
33
34     ALARM_DRIVER_state = STATE(ALARM_DRIVER_waiting);
35 }
36 STATE_define(ALARM_DRIVER_waiting)
37 {
38     alarmDriver_state_id = alarmDriver_waiting;
39 }

```

Figure 13: alarmDriver.c



```

1  #ifndef _ALARM_DRIVER_H_
2  #define _ALARM_DRIVER_H_
3
4  #include "stateMachine.h"
5
6  enum {
7      alarmDriver_alarmON,
8      alarmDriver_alarmOFF,
9      alarmDriver_waiting
10 }alarmDriver_state_id;
11
12 void alarmDriver_init();
13
14 STATE_define(ALARM_DRIVER_alarmON);
15 STATE_define(ALARM_DRIVER_alarmOFF);
16 STATE_define(ALARM_DRIVER_waiting);
17
18 extern void (*ALARM_DRIVER_state) ();
19
20 #endif

```

Figure 14: alarmDriver.c

```

1  #include "alarmMonitor.h"
2  #include "driver.h"
3
4  void (*ALARM_MONITOR_state) ();
5  alarmDuration = 600000;
6  void highPressureDetected()
7  {
8      DPRINTF("high pressure detected: starting alarm in progress.....\n");
9      ALARM_MONITOR_state = STATE(ALARM_MONITOR_alarmON);
10 }
11 STATE_define(ALARM_MONITOR_alarmON)
12 {
13     alarmMonitor_state_id = alarmMonitor_alarmON;
14     DPRINTF("ALARM_MONITOR_alarmON state.....\n");
15
16     startTimer();
17
18
19     ALARM_MONITOR_state = STATE(ALARM_MONITOR_waiting);
20 }
21 STATE_define(ALARM_MONITOR_alarmOFF)
22 {
23     alarmMonitor_state_id = alarmMonitor_alarmOFF;
24     DPRINTF("ALARM_MONITOR_alarmOFF state.....\n");
25
26     stopTimer();
27
28     ALARM_MONITOR_state = STATE(ALARM_MONITOR_alarmOFF);
29 }
30 STATE_define(ALARM_MONITOR_waiting)
31 {
32     alarmMonitor_state_id = alarmMonitor_waiting;
33     DPRINTF("ALARM_MONITOR_waiting state.....\n");
34     Delay(alarmDuration);
35     ALARM_MONITOR_state = STATE(ALARM_MONITOR_alarmOFF);
36     ALARM_MONITOR_state();
37 }

```

Figure 15: alarmMonitor.c

```

1  #ifndef _ALARM_MONITOR_H_
2  #define _ALARM_MONITOR_H_
3
4  #include "stateMachine.h"
5
6  enum {
7      alarmMonitor_alarmON,
8      alarmMonitor_alarmOFF,
9      alarmMonitor_waiting
10 }alarmMonitor_state_id;
11
12 STATE_define(ALARM_MONITOR_alarmON);
13 STATE_define(ALARM_MONITOR_alarmOFF);
14 STATE_define(ALARM_MONITOR_waiting);
15
16 extern void (*ALARM_MONITOR_state) ();
17
18 #endif

```

Figure 16: alarmMonitor.h

```

1  #include "mainAlg.h"
2  #include "platform_Types.h"
3
4  static uint32 pVal = 0;
5  static uint32 pThreshold = 20;
6  void (*MAIN_ALG_state) ();
7
8  void readPressureValue(int pValue)
9  {
10     pVal = pValue;
11     DPRINTF("mainAlg:.....pressure value=%d...../n",pVal);
12 }
13
14 STATE_define(MAIN_ALG_highPressureDetect)
15 {
16     mainAlg_state_id = mainAlg_highPressireDetect;
17
18     if(pVal >= pThreshold)
19         highPressureDetected();
20     DPRINTF("MAIN_ALG_highPressureDetect...../n",pVal);
21     MAIN_ALG_state = STATE(MAIN_ALG_highPressureDetect);
22 }
23

```

Figure 17: mainAlg.c

```
1  #ifndef _MAIN_ALG_H_
2  #define _MAIN_ALG_H_
3
4  #include "stateMachine.h"
5
6  enum {
7      mainAlg_highPressureDetect
8  }mainAlg_state_id;
9
10
11  STATE_define(MAIN_ALG_highPressureDetect);
12
13  extern void (*MAIN_ALG_state) ();
14
15  #endif
```

Figure 18: mainAlg.h

```

1  #include "pressureDriver.h"
2  #include "driver.h"
3  #include "platform_Types.h"
4
5  void (*PRESSURE_state) ();
6  static uint32 pVal = 0;
7
8  void pressure_init()
9  {
10     DPRINTF("PRESSURE INIT...\n");
11 }
12
13 STATE_define(PRESSURE_reading)
14 {
15     pressureDriver_state_id = pressure_reading;
16
17     //reading from sensor
18     pVal = getPressureVal();
19
20     DPRINTF("pressure_reading state, pVal = %d\n",pVal);
21
22     //sending the pressure signal
23     readPressureValue(pVal);
24
25     //next state
26     PRESSURE_state = STATE(PRESSURE_waiting);
27 }
28
29
30
31 STATE_define(PRESSURE_waiting)
32 {
33     pressureDriver_state_id = pressure_waiting;
34
35     //Delay
36     Delay(200000);
37
38     //next state
39     PRESSURE_state = STATE(PRESSURE_reading);
40 }
41

```

Figure 19: pressureDriver.c

```

1  #ifndef _PRESSURE_DRIVER_H_
2  #define _PRESSURE_DRIVER_H_
3
4  #include "stateMachine.h"
5
6  enum {
7      pressure_reading,
8      pressure_waiting
9  }pressureDriver_state_id;
10
11 void pressure_init();
12
13 STATE_define(PRESSURE_reading);
14 STATE_define(PRESSURE_waiting);
15
16 extern void (*PRESSURE_state) ();
17
18 #endif

```

Figure 20: pressureDriver.h

```

1  #ifndef _STATEMAHCINE_H_
2  #define _STATEMAHCINE_H_
3
4  #define STATE_define(_stateFunc_)  void ST_##_stateFunc_(void)
5  #define STATE(_stateFunc_)  ST_##_stateFunc_
6
7  #define DPRINTF(...)  { /*fflush(stdin);\
8                        fflush(stdout);\
9                        printf(__VA_ARGS__);\
10                       fflush(stdin);\
11                       fflush(stdout);*/}
12
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <platform_Types.h>
16
17 //states connection
18 void readPressureValue(int pValue);
19 void highPressureDetected();
20 void startTimer();
21 void stopTimer();
22
23 #endif

```

Figure 21: stateMachine.h

```

4  *      Created on: 27 Jan 2023
5  *      Author:      Ahmed Sameh
6  */
7  #ifndef PLATFORM_TYPES_H_
8  #define PLATFORM_TYPES_H_
9  #ifndef boolean
10 #define boolean unsigned char
11 #endif
12 #define true      1
13 #define false     0
14 #ifndef FALSE
15 #define FALSE     (boolean)false
16 #endif
17 #ifndef TRUE
18 #define TRUE      (boolean>true
19 #endif
20 typedef char sint8;
21 typedef unsigned char uint8;
22 typedef char char_t;
23 typedef short sint16;
24 typedef unsigned short uint16;
25 typedef long sint32;
26 typedef unsigned long uint32;
27 typedef long long sint64;
28 typedef unsigned long long uint64;
29 typedef volatile char vint8_t;
30 typedef volatile unsigned char vuint8_t;
31
32 typedef volatile short vint16_t;
33 typedef volatile unsigned short vuint16_t;
34
35 typedef volatile long vint32_t;
36 typedef volatile unsigned long vuint32_t;
37
38 typedef volatile long long vint64_t;
39 typedef volatile unsigned long long vuint64_t;
40
41 typedef float float32;
42 typedef double float64;
43
44
45 #endif /* PLATFORM_TYPES_H_ */
46

```

Figure 22: platformTypes.h

## Makefile

```
1  # created by Eng Ahmed Sameh in Mastering embedded systems diploma unit 5 project 1
2  #CC stands for C compiler as it is used to change compiler toolchain
3  CC=arm-none-eabi-
4  #CFlags means the option used while compiling
5  CFLAGS=-mcpu=cortex-m3 -mthumb -gdwarf-2
6  #INCS stands for include as we should include all files in the folders
7  INCS=-I .
8  #Libs stands for libraries as it is used to include libraries made before
9  LIBS=
10
11  SRC=$(wildcard *.c)
12  OBJ=$(SRC:.c=.o)
13  As=$(wildcard *.s)
14  AsOBJ=$(As:.s=.o)
15
16  ProjectName= FirstTermProject1
17
18  # Add an @ before a command to stop it from being printed
19  all: $(ProjectName).bin
20      @echo '*****Done*****'
21
22  # the 2> log is used to forward error messages into a log file so that the terminal stays clean
23  # we can't insert comments besides the commands or the target: dependency, comments must be in sperated lines
24  # %.o meany any file.o, $@ means target file, $< means first dependency file
25  %.o: %.s
26      $(CC)as.exe $(CFLAGS) $< -o $@ 2> log
27
28  %.o: %.c
29      $(CC)gcc.exe -c $(CFLAGS) $(INCS) $< -o $@
30
31  $(ProjectName).elf: $(OBJ) $(AsOBJ)
32      $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $@ -Map=Map_file.map
33
34  $(ProjectName).bin: $(ProjectName).elf
35      $(CC)objcopy.exe -O binary $< $@
36
37  clean_all:
38      rm *.o *.elf *.bin
39
40  clean:
41      rm *.elf *.bin
```

Figure 23: Makefile



## Linker file

```
1  /*
2      Author: Ahmed Sameh Elshahed
3      project: FirstTerm project 1
4      file name: linker_script.ld
5  */
6
7  MEMORY
8  {
9      flash(RX): ORIGIN = 0x08000000, LENGTH = 128K
10     sram(RWX): ORIGIN = 0x20000000, LENGTH = 20K
11 }
12
13 SECTIONS
14 {
15     .text : {
16         *(.vectors*)
17         *(.text*)
18         *(.rodata*)
19         _E_text = .;
20     } > flash
21     .data : {
22         _S_data = .;
23         *(.data*)
24         _E_data = .;
25     } > sram AT > flash
26     .bss : {
27         _S_bss = .;
28         *(.bss*)
29         . = ALIGN(4);
30         _E_bss = .;
31
32         . = ALIGN(4);
33         . = . + 0x1000;
34         _stack_top = .;
35     } > sram
36 }
```

Figure 24: linker\_script.ld

## Startup file

```
1 // created by Ahmed Sameh in Mastering embedded systems diploma
2 #include <platform_Types.h>
3 extern int main(void);
4 void Reset_Handler(void);
5 void Default_Handler(){
6     Reset_Handler();
7 }
8 void NMI_Handler(void) __attribute__((weak, alias("Default_Handler")));
9 void H_fault_Handler(void) __attribute__((weak, alias("Default_Handler")));
10 void MM_fault_Handler(void) __attribute__((weak, alias("Default_Handler")));
11 void Bus_fault(void) __attribute__((weak, alias("Default_Handler")));
12 void Usage_fault_Handler(void) __attribute__((weak, alias("Default_Handler")));
13 //reserve stack
14 static uint32 Stack_top[256];
15 void (* const vectors_arr[])(void) __attribute__((section(".vectors"))) = {
16     (void(*) (void)) ((uint32)Stack_top + (uint32)sizeof(Stack_top)),
17     &Reset_Handler,
18     &NMI_Handler,
19     &H_fault_Handler,
20     &MM_fault_Handler,
21     &Bus_fault,
22     &Usage_fault_Handler };
23 extern uint32 _E_text;
24 extern uint32 _S_data;
25 extern uint32 _E_data;
26 extern uint32 _S_bss;
27 extern uint32 _E_bss;
28 void Reset_Handler(void) {
29     // copy data section from flash to ram
30     uint32 DataSize = (uint8 *)&_E_data - (uint8 *)&_S_data;
31     uint8 *P_src = (uint8 *)&_E_text;
32     uint8 *P_dst = (uint8 *)&_S_data;
33     uint32 i;
34     for(i = 0; i < DataSize; i++)
35         *((uint8 *)P_dst++) = *((uint8 *)P_src++);
36     // initialize the bss section with zeros
37     uint32 BssSize = (uint8 *)&_E_bss - (uint8 *)&_S_bss;
38     P_dst = (uint8 *)&_S_bss;
39     for(i = 0; i < BssSize; i++)
40         *((uint8 *)P_dst++) = (uint8)0;
41     // jump to main
42     main(); }
```

Figure 25: startup.c

## Map\_file.map

7		0x1	alarmDriver.o
8	alarmMonitor_state_id		
9		0x1	alarmMonitor.o
10	mainAlg_state_id	0x1	main.o
11	pressureDriver_state_id		
12		0x1	main.o
13	MAIN_ALG_state	0x4	mainAlg.o
14	PRESSURE_state	0x4	pressureDriver.o
15	ALARM_MONITOR_state		
16		0x4	alarmMonitor.o
17			
18	Memory Configuration		
19			
20	Name	Origin	Length
21	flash	0x08000000	0x00020000
22	sram	0x20000000	0x00005000
23	*default*	0x00000000	0xffffffff
24			
25	Linker script and memory map		
26			
27			
28	.text	0x08000000	0x4f0
29	*(.vectors*)		
30	.vectors	0x08000000	0x1c startup.o
31		0x08000000	vectors_arr
32	*(.text*)		
33	.text	0x0800001c	0xd0 alarmDriver.o
34		0x0800001c	alarmDriver_init
35		0x0800002c	startTimer
36		0x08000050	stopTimer
37		0x08000074	ST_ALARM_DRIVER_alarmON
38		0x080000a4	ST_ALARM_DRIVER_alarmOFF
39		0x080000d4	ST_ALARM_DRIVER_waiting
40	.text	0x080000ec	0xb8 alarmMonitor.o
41		0x080000ec	highPressureDetected
42		0x08000108	ST_ALARM_MONITOR_alarmON
43		0x08000134	ST_ALARM_MONITOR_alarmOFF
44		0x08000160	ST_ALARM_MONITOR_waiting
45	.text	0x080001a4	0x10c driver.o
46		0x080001a4	Delay
47		0x080001c8	getPressureVal
48		0x080001e0	Set_Alarm_actuator
49		0x08000230	GPIO_INITIALIZATION
50	.text	0x080002b0	0x98 main.o
51		0x080002b0	setup
52		0x0800030c	main
53	.text	0x08000348	0x64 mainAlg.o
54		0x08000348	readPressureValue
55		0x08000368	ST_MAIN_ALG_highPressureDetect
56	.text	0x080003ac	0x88 pressureDriver.o
57		0x080003ac	pressure_init
58		0x080003b8	ST_PRESSURE_reading
59		0x08000400	ST_PRESSURE_waiting
60	.text	0x08000434	0xbc startup.o
61		0x08000434	Bus_fault
62		0x08000434	MM_fault_Handler
63		0x08000434	Usage_fault_Handler
64		0x08000434	H_fault_Handler
65		0x08000434	Default_Handler

Figure 26: Map\_file.map

## Symbol Table

1	20000410	B	_E_bss
2	20000008	D	_E_data
3	080004f0	T	_E_text
4	20000008	B	_S_bss
5	20000000	D	_S_data
6	20001410	B	_stack_top
7	20001410	B	ALARM_DRIVER_state
8	2000141c	B	ALARM_MONITOR_state
9	0800001c	T	alarmDriver_init
10	20001414	B	alarmDriver_state_id
11	20000000	D	alarmDuration
12	20001418	B	alarmMonitor_state_id
13	08000434	W	Bus_fault
14	08000434	T	Default_Handler
15	080001a4	T	Delay
16	080001c8	T	getPressureVal
17	08000230	T	GPIO_INITIALIZATION
18	08000434	W	H_fault_Handler
19	080000ec	T	highPressureDetected
20	0800030c	T	main
21	20001424	B	MAIN_ALG_state
22	20001420	B	mainAlg_state_id
23	08000434	W	MM_fault_Handler
24	08000434	W	NMI_Handler
25	080003ac	T	pressure_init
26	20001428	B	PRESSURE_state
27	20001421	B	pressureDriver_state_id
28	20000004	d	pThreshold
29	20000008	b	pVal
30	2000000c	b	pVal
31	08000348	T	readPressureValue
32	08000440	T	Reset_Handler
33	080001e0	T	Set_Alarm_actuator
34	080002b0	T	setup
35	080000a4	T	ST_ALARM_DRIVER_alarmOFF
36	08000074	T	ST_ALARM_DRIVER_alarmON
37	080000d4	T	ST_ALARM_DRIVER_waiting
38	08000134	T	ST_ALARM_MONITOR_alarmOFF
39	08000108	T	ST_ALARM_MONITOR_alarmON
40	08000160	T	ST_ALARM_MONITOR_waiting
41	08000368	T	ST_MAIN_ALG_highPressureDetect
42	080003b8	T	ST_PRESSURE_reading
43	08000400	T	ST_PRESSURE_waiting
44	20000010	b	Stack_top
45	0800002c	T	startTimer
46	08000050	T	stopTimer
47	08000434	W	Usage_fault_Handler
48	08000000	T	vectors_arr
49			

Figure 27: symbol Table

## Simulation Results:

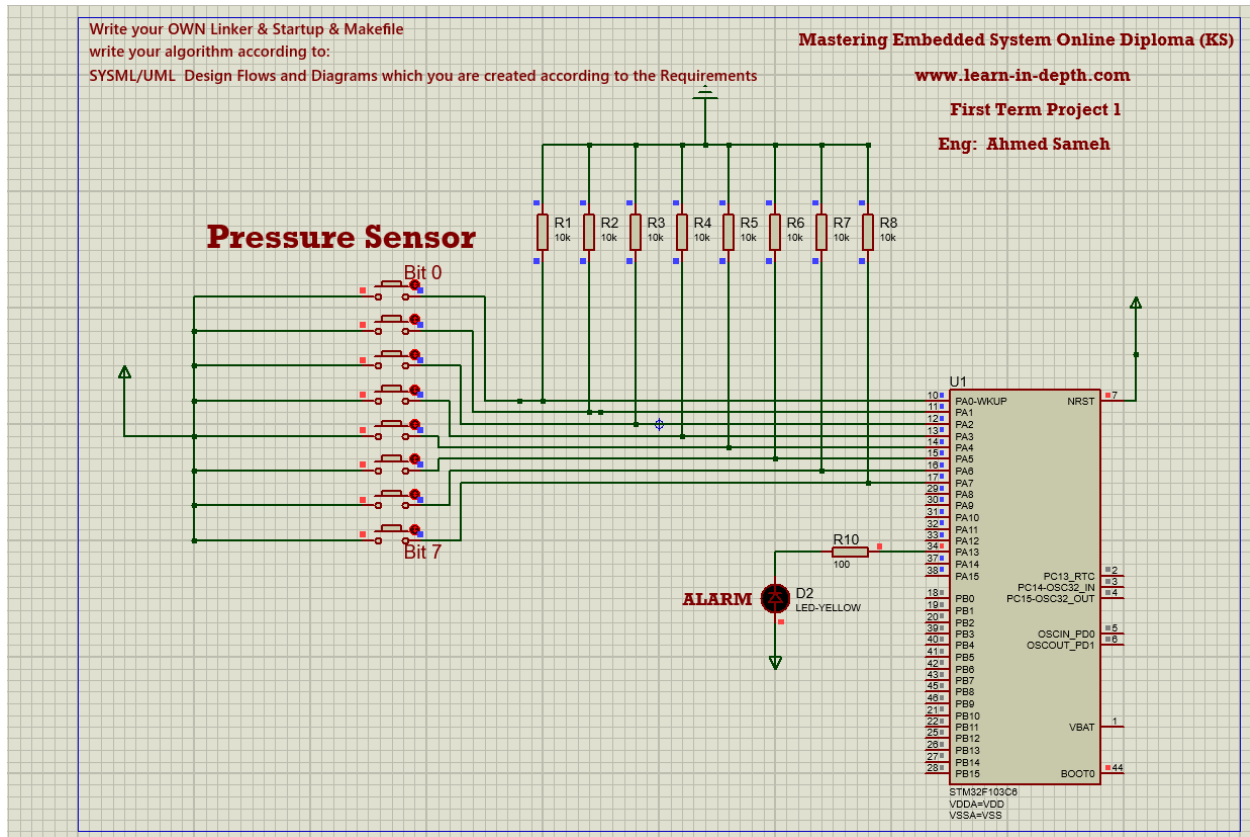


Figure 28: simulation results when pressure is less than 20

**SYSML/UML Design Flows and Diagrams which you are created according to the Requirements**

[www.learn-in-depth.com](http://www.learn-in-depth.com)

**Eng: Ahmed Sameh**

