

Mastering embedded systems online diploma

www.learn-in-depth.com

First term (Project 1)

Eng. [Omar Mahmoud Elmasri](#)

My profile:

learn-in-depth/OmarElmasri

Table of Contents

1	Brief intro.....	3
2	Case Study.....	3
3	Requirements Diagram.....	3
4	System Analysis.....	3
4.1	Use Case Diagram.....	4
4.2	Activity Diagram.....	4
4.3	Sequence Diagram.....	5
5	System Design.....	6
5.1	State machine.....	7
6	Source Code.....	8
7	Simulation.....	9

Table of Figures

Figure 1:	Requirements diagram.....	3
Figure 2:	Use case diagram.....	4
Figure 3:	Activity diagram.....	4
Figure 4:	Sequence diagram.....	5
Figure 5:	System design using ttool.....	6
Figure 6:	Main Algorithm states.....	7
Figure 7:	Main Algorithm .c file.....	8
Figure 8:	Main Algorithm .h file.....	9
Figure 9:	pressure below threshold of 20, LED is off.....	9
Figure 10:	Pressure is 24 above threshold of 20, LED is on.....	10
Figure 11:	Pressure is below threshold, yet, LED is on due to delay.....	11

1 Brief intro

Pressure Alarm system, project 1. Learn in depth diploma.

2 Case Study

System shall read pressure sensor output.

System shall generate output signal to alarm actuator.

System shall maintain alarm signal for specified time.

3 Requirements Diagram

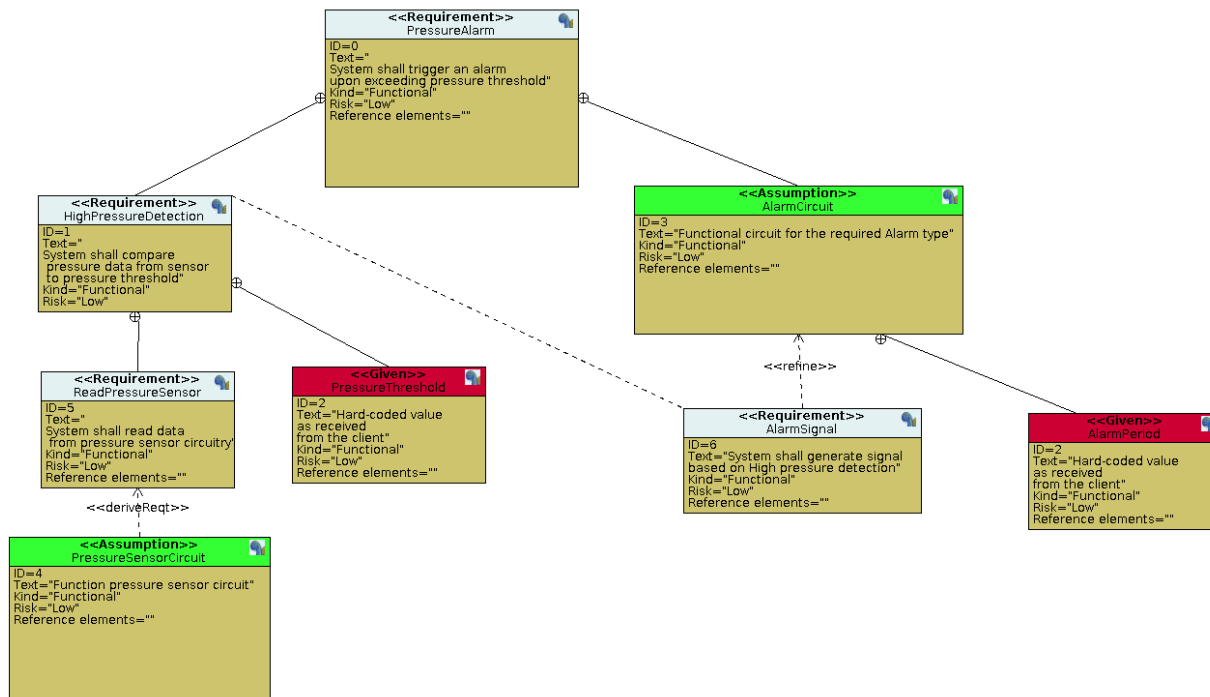


Figure 1: Requirements diagram

4 System Analysis

4.1 Use Case Diagram

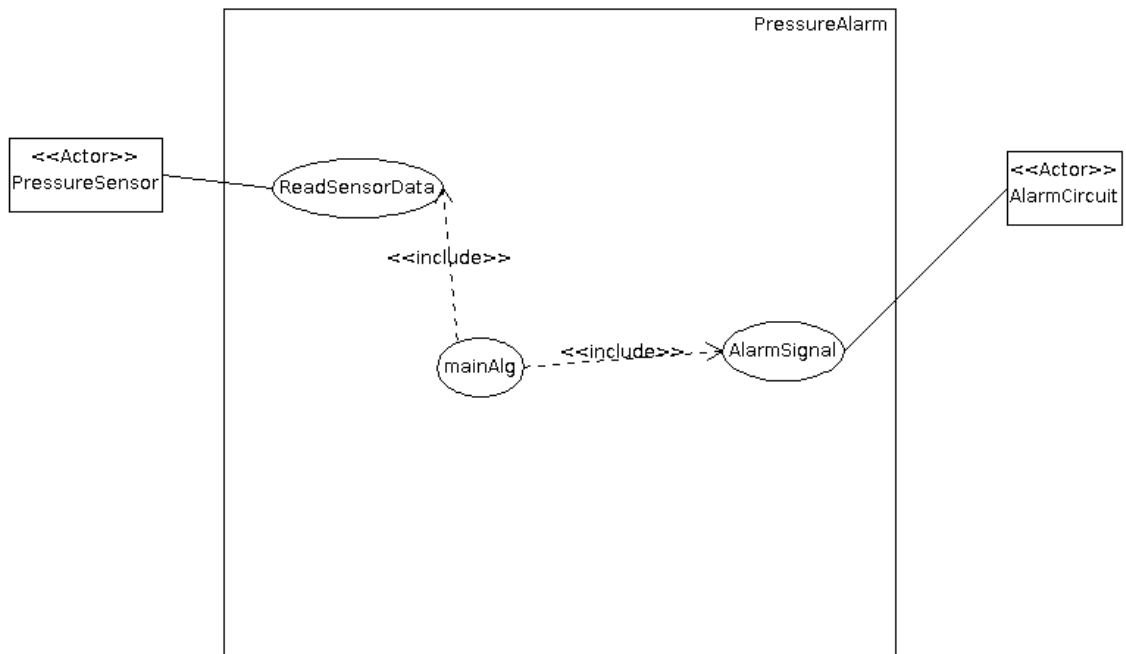


Figure 2: Use case diagram

4.2 Activity Diagram

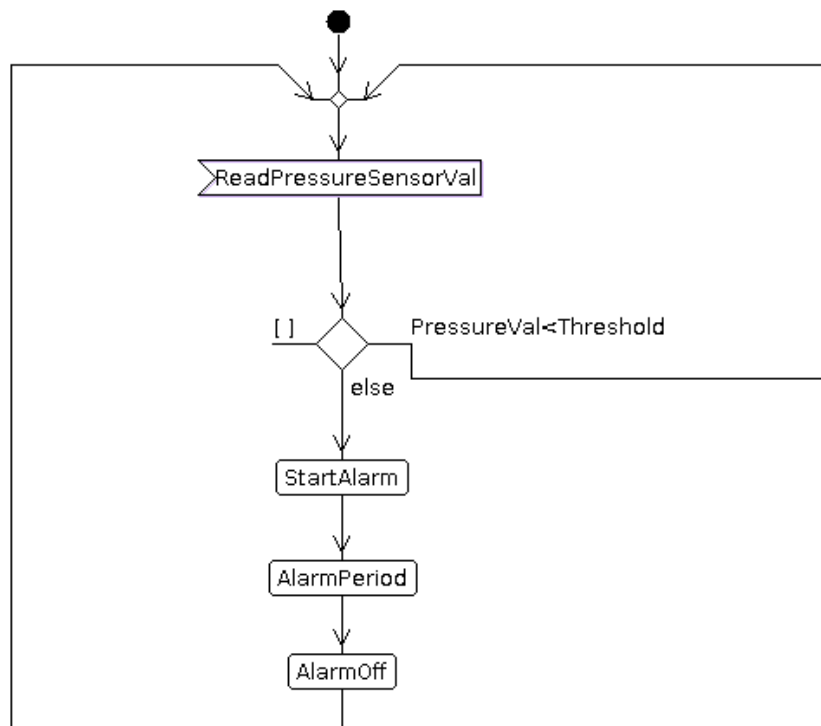


Figure 3: Activity diagram

4.3 Sequence Diagram

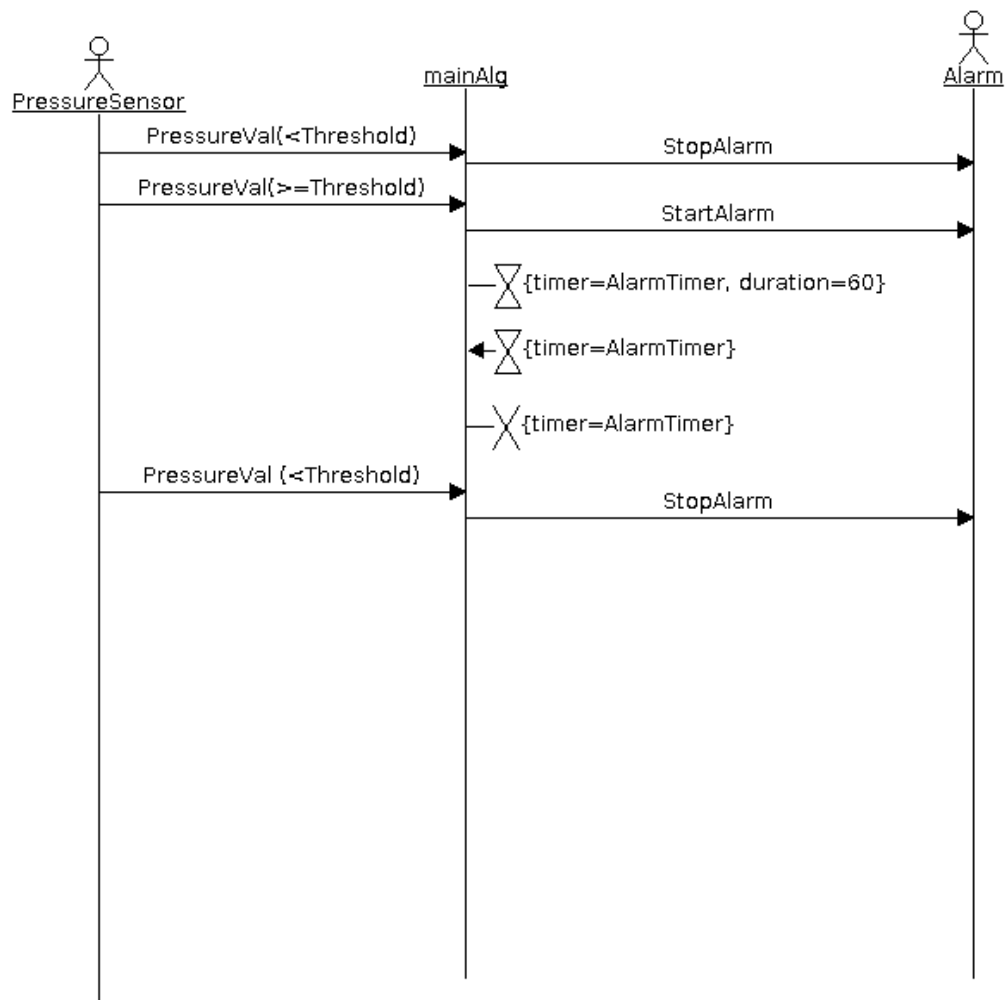


Figure 4: Sequence diagram

5 System Design

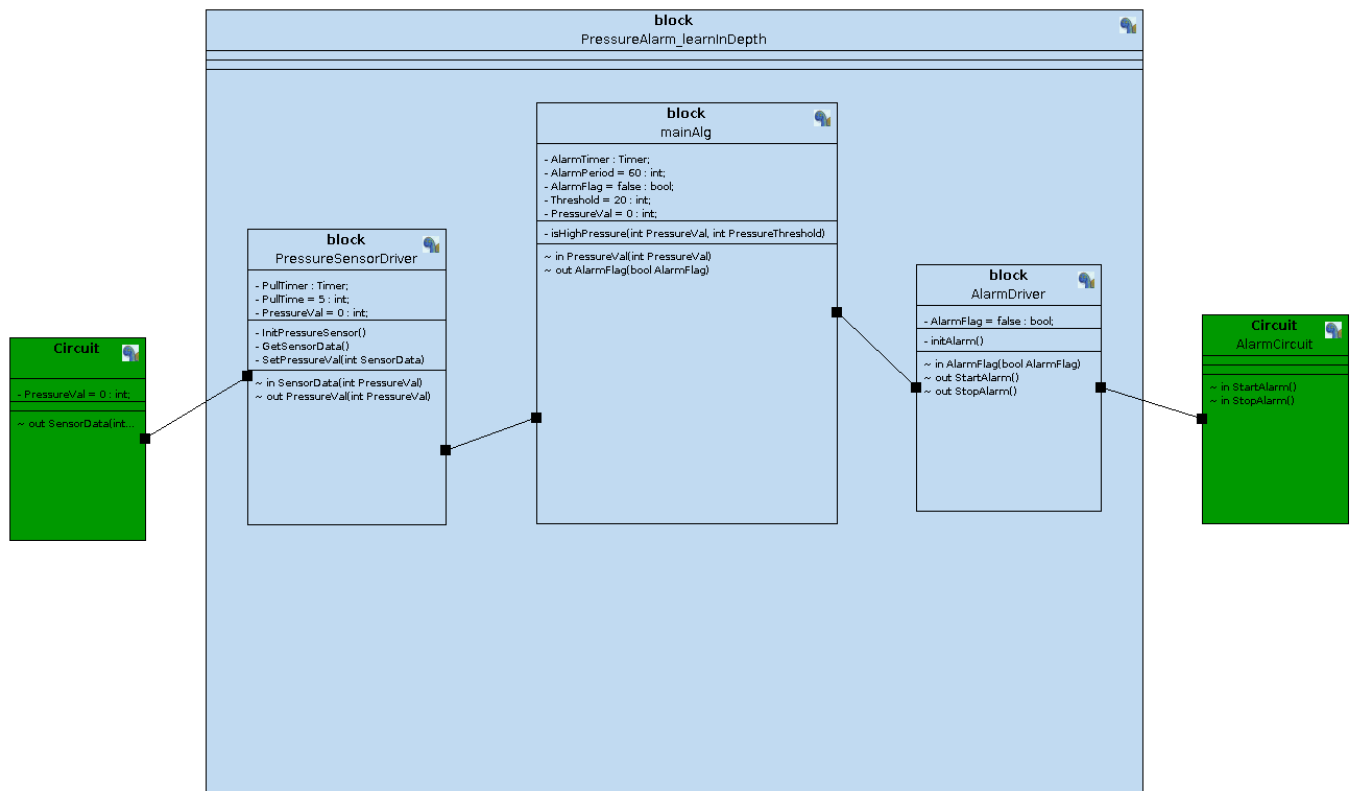


Figure 5: System design using ttool

5.1 State machine

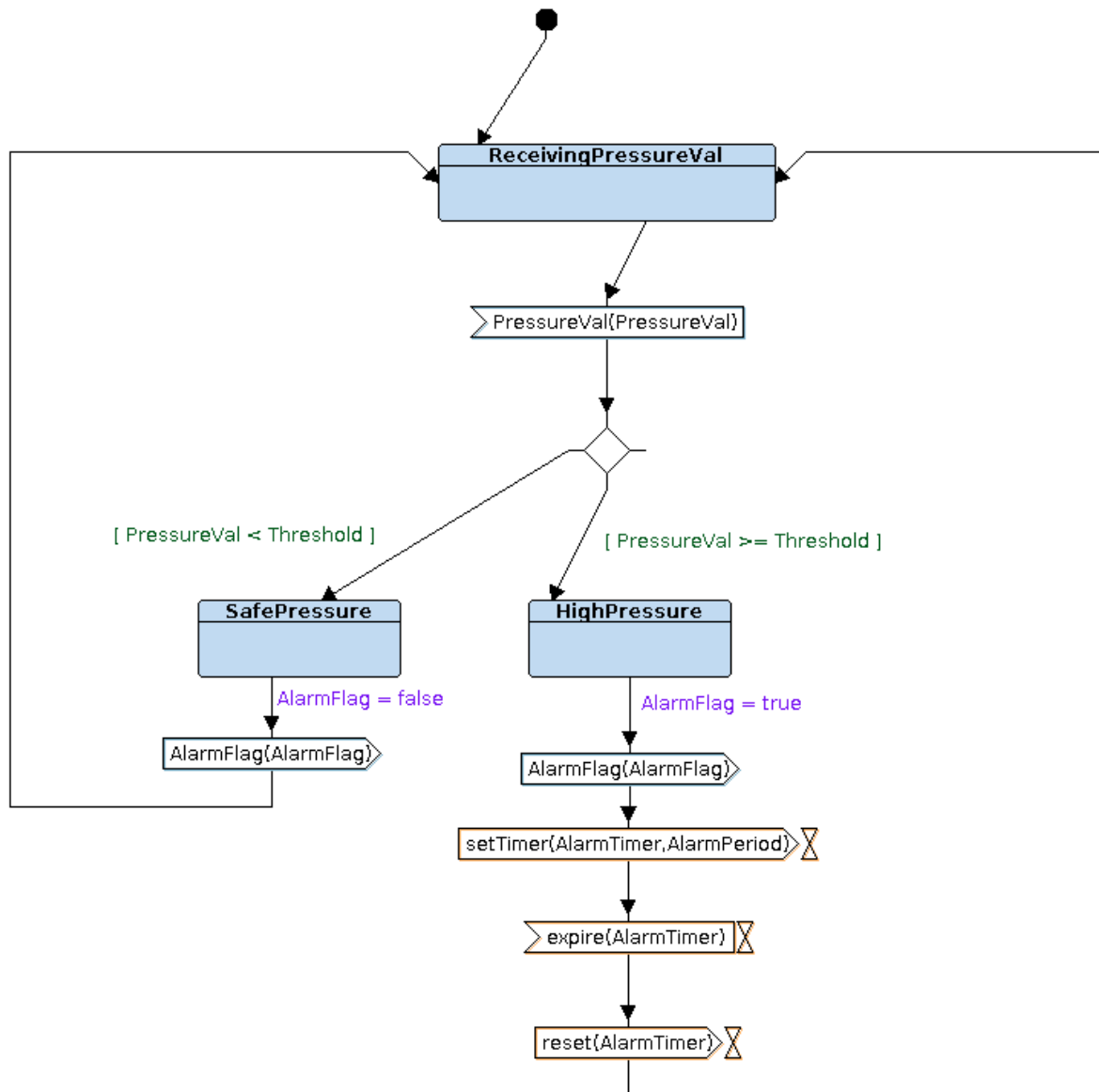


Figure 6: Main Algorithm states

[Also, follow the link for state machine simulation](#)

since driver code was provided, hence, drivers state machine modules are irrelevant.

6 Source Code

```
Unit4 > lesson3 > STM32F103C8-BluePill > mainAlg.c > init(void)
22
23 int AlarmFlag = 0;
24 void (*state_ptr_main)(void);
25
26 void init(void) {
27     GPIO_INITIALIZATION();
28     state_ptr_main = ReceivingPressureVal;
29     AlarmFlag = 0;
30 }
31 //d]: implement states according to design
32
33 //d]: high pressure state
34
35 /// represent state where pressure value is above or equal threshold
36 STATE(HighPressure) {
37     /// start alarm.
38     AlarmFlag = 0; /// reverse flag to match reverse polarity provided by the user.
39     Set_Alarm_actuator(AlarmFlag); /// using the driver provided.
40
41     /// start timer
42     Delay(ALARM_PERIOD);
43
44     /// switch state
45     state_ptr_main = ReceivingPressureVal;
46 }
47 //d]: safe pressure state
48 /// represent state where pressure value is below threshold
49 STATE(SafePressure) {
50     /// stop alarm
51     AlarmFlag = 1; /// reverse flag to match reverse polarity provided by the user.
52     Set_Alarm_actuator(AlarmFlag); /// using the driver provided
53
54     /// switch state
55     state_ptr_main = ReceivingPressureVal;
56 }
57 //d]: receiving pressure state
58 /// represent state where pressure value is being measured
59 STATE(ReceivingPressureVal) {
60     /// read pressure value
61     int pressure = PressureSensor_set_pressure();
62     Delay(PULL_TIME);
63
64     /// switch state
65     state_ptr_main =
66         (pressure >= PRESSURE_THRESHOLD) ? HighPressure : SafePressure;
67 }
68
```

Figure 7: Main Algorithm .c file

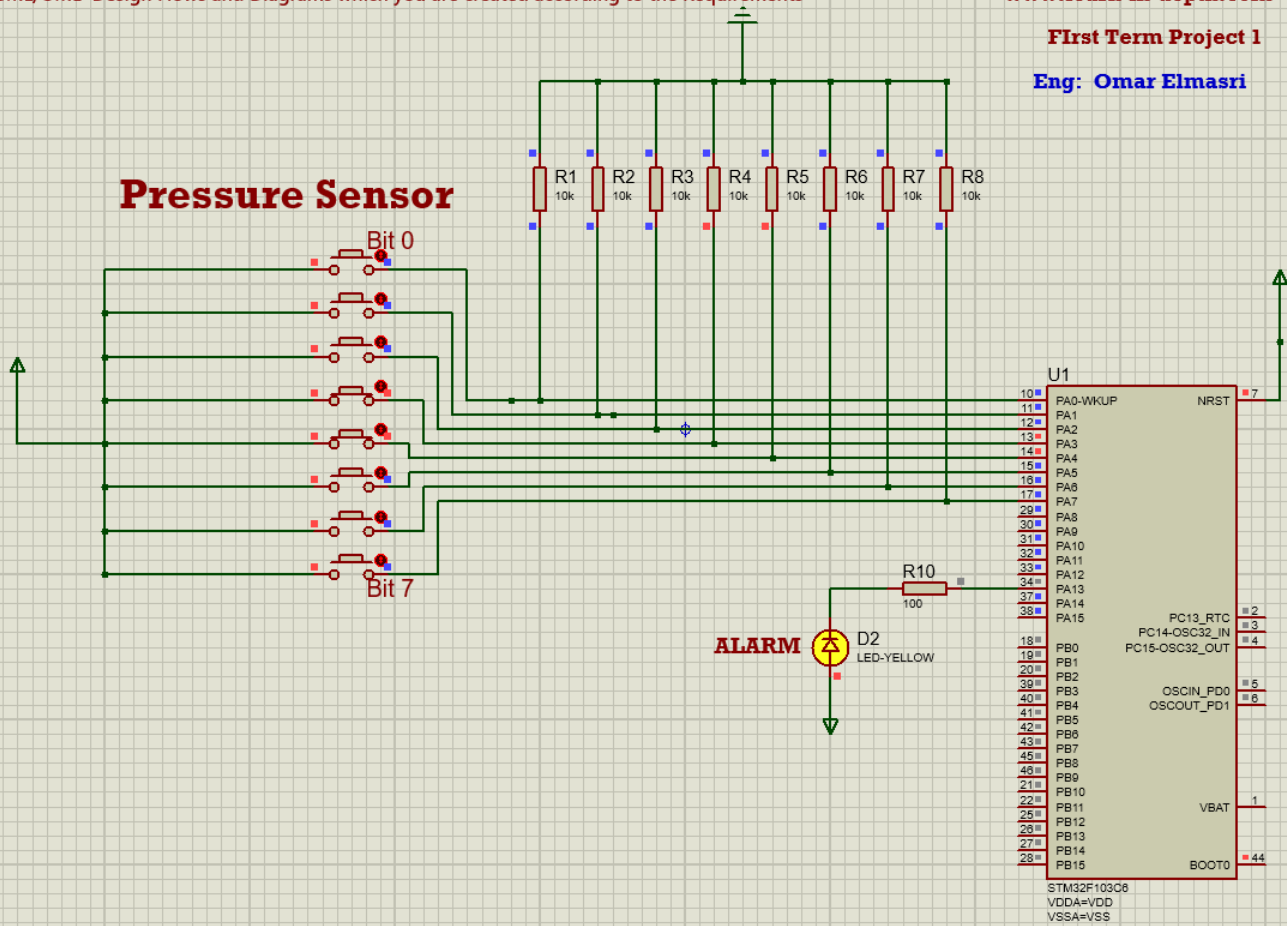


Figure 10: Pressure is 24 above threshold of 20, LED is on

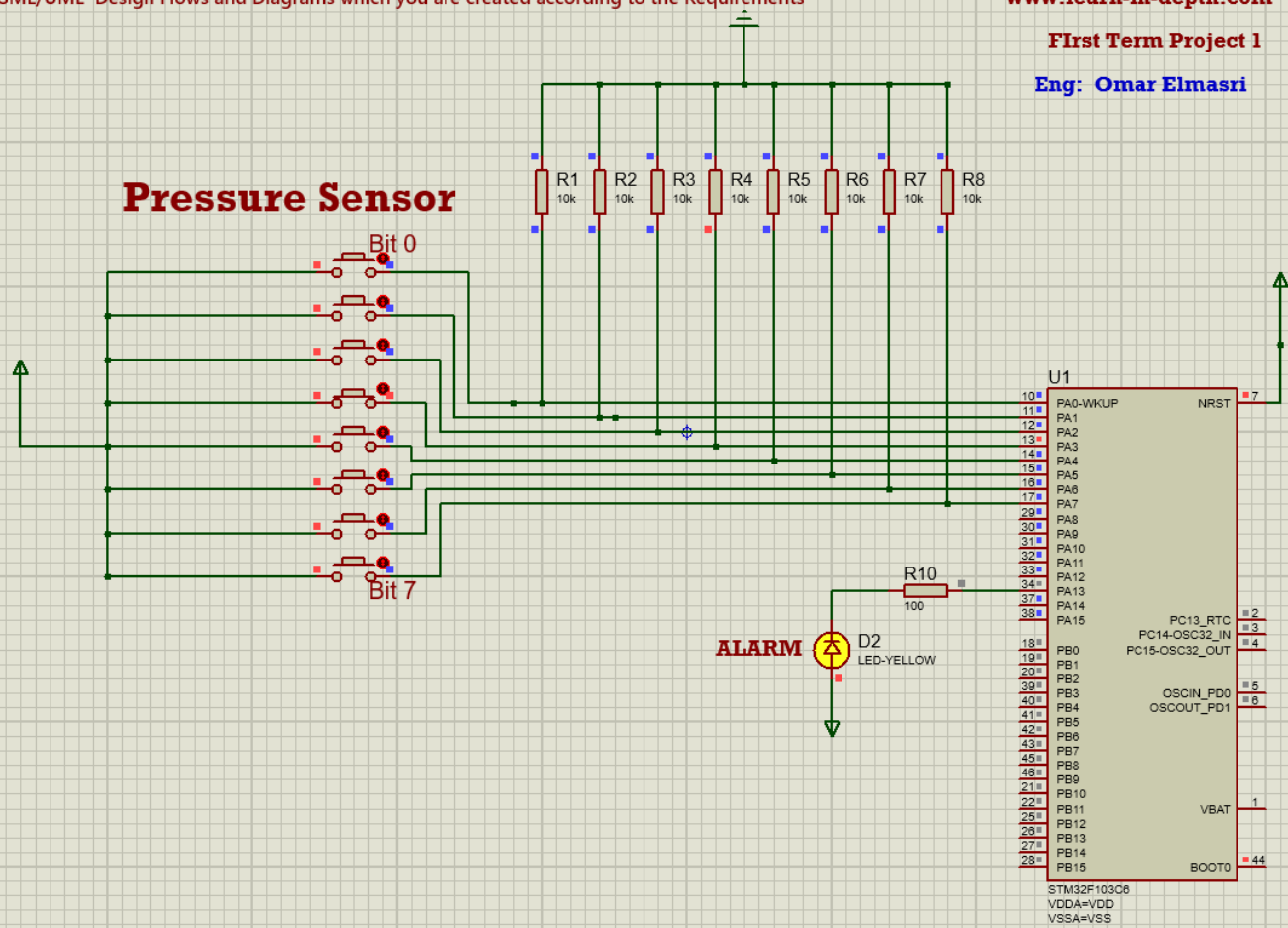


Figure 11: Pressure is below threshold, yet, LED is on due to delay