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

Table of Figures

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

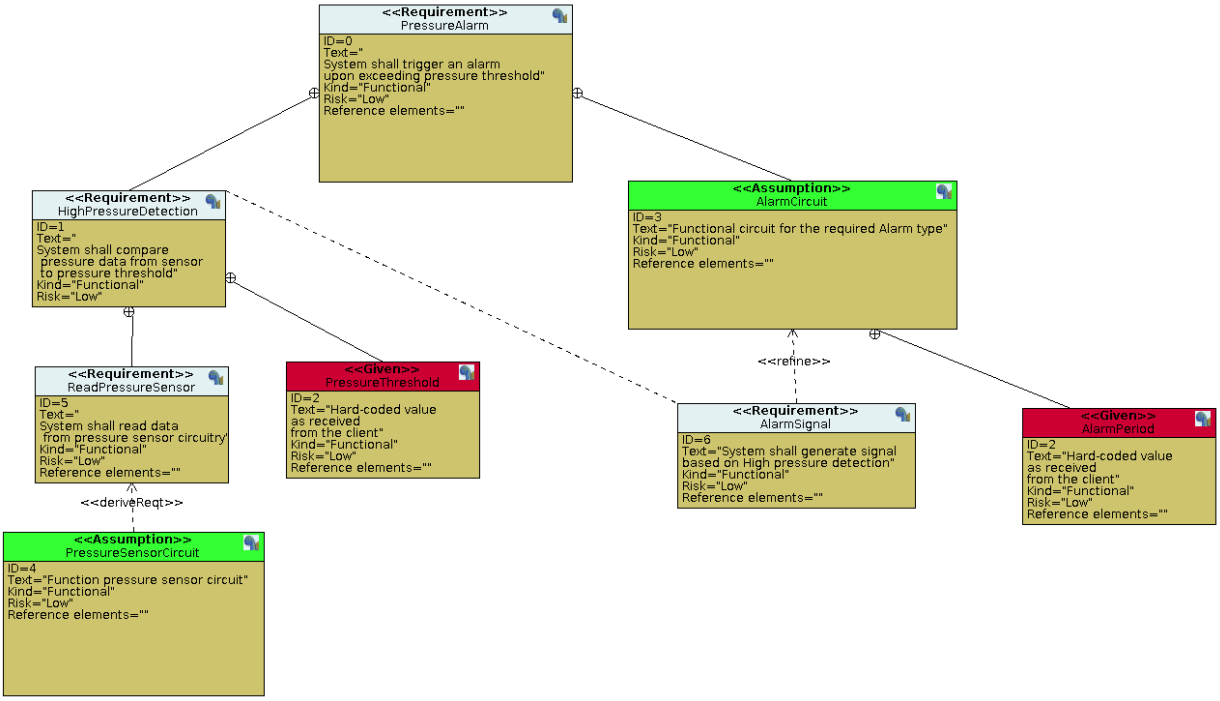


Figure 1: Requirements diagram

4 System Analysis

4.1 Use Case Diagram

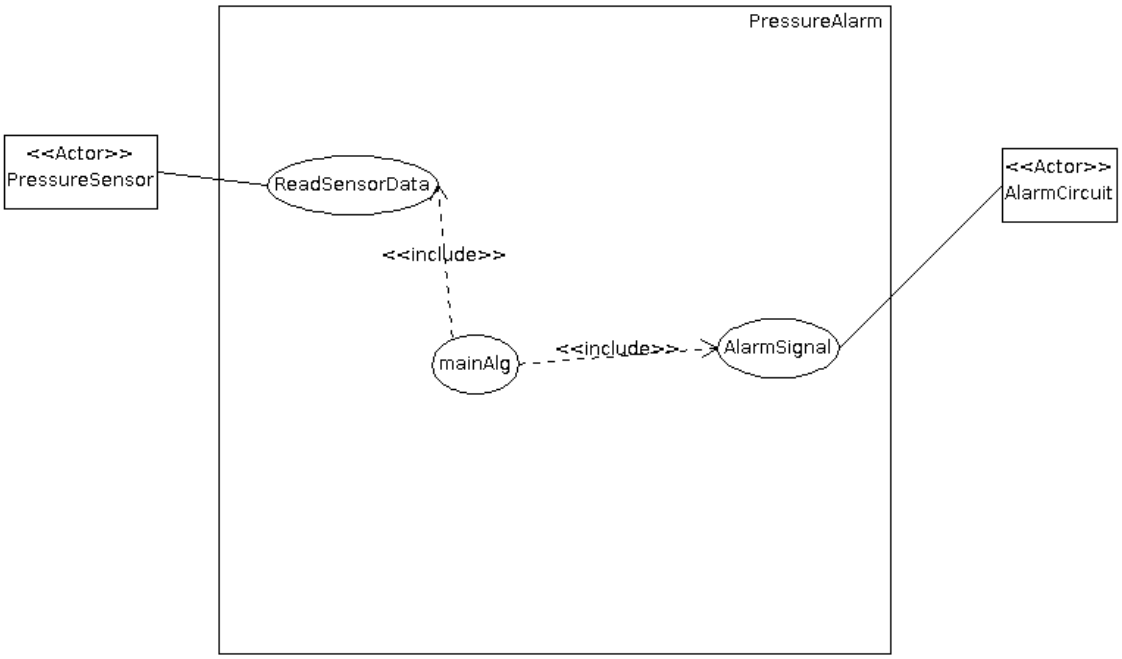


Figure 2: Use case diagram

4.2 Activity Diagram

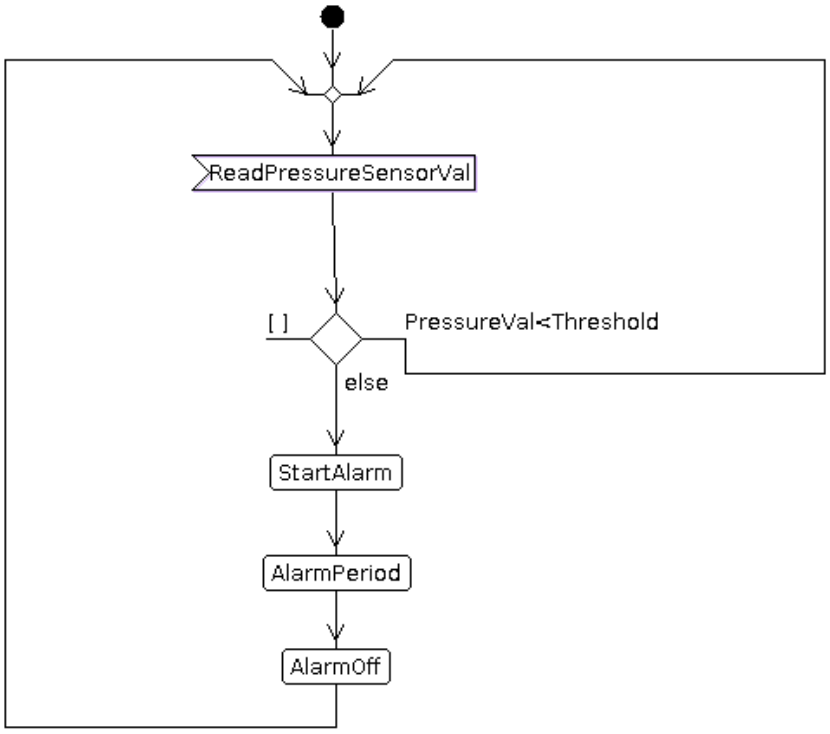


Figure 3: Activity diagram

4.3 Sequence Diagram

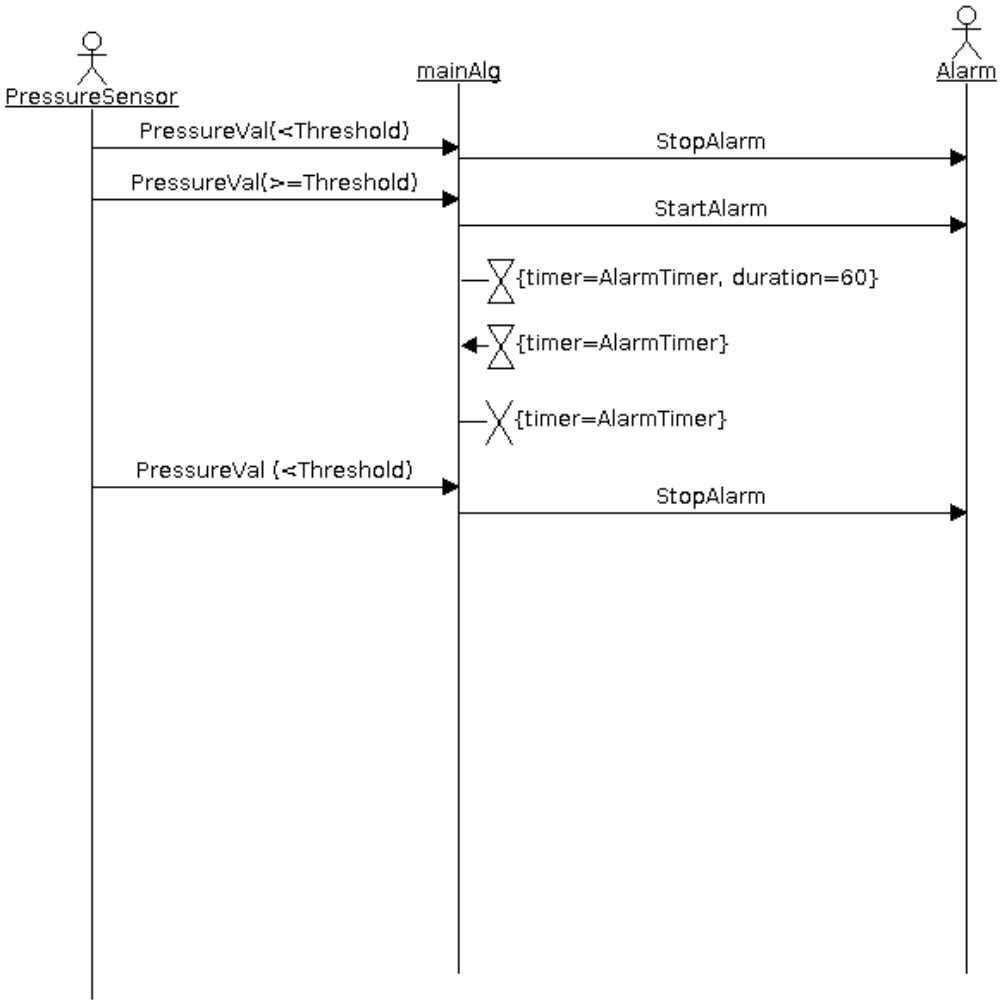


Figure 4: Sequence diagram

5 System Design

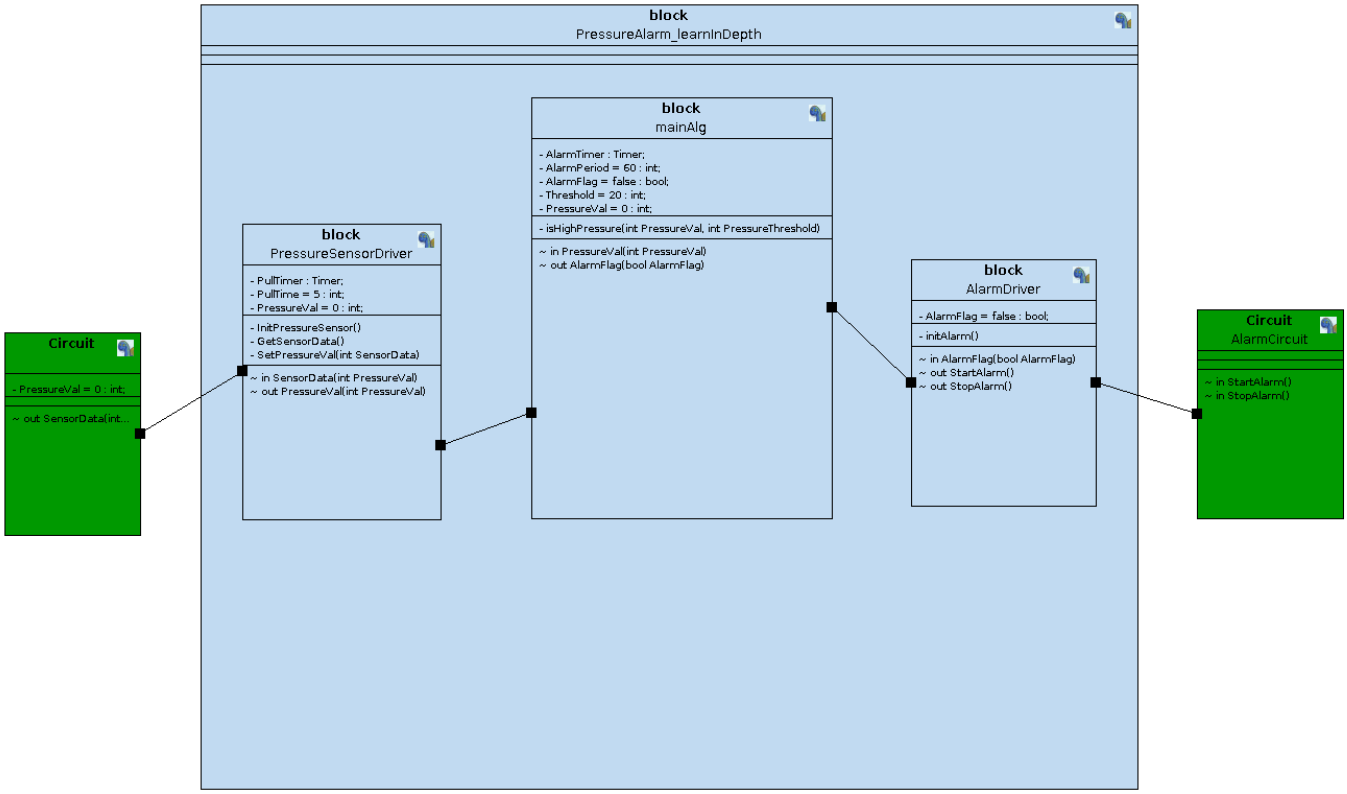


Figure 5: System design using ttool

5.1 State machine

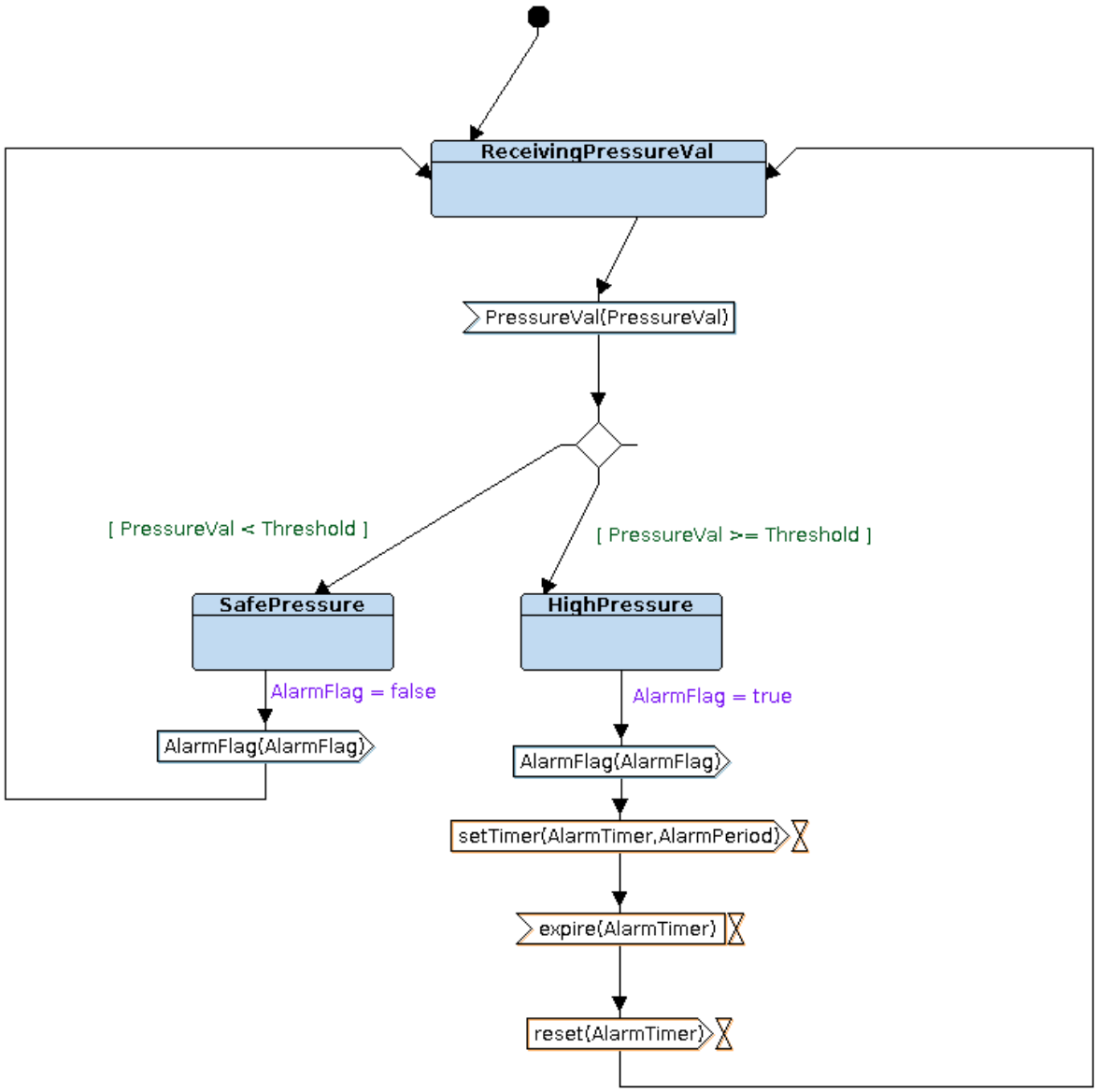
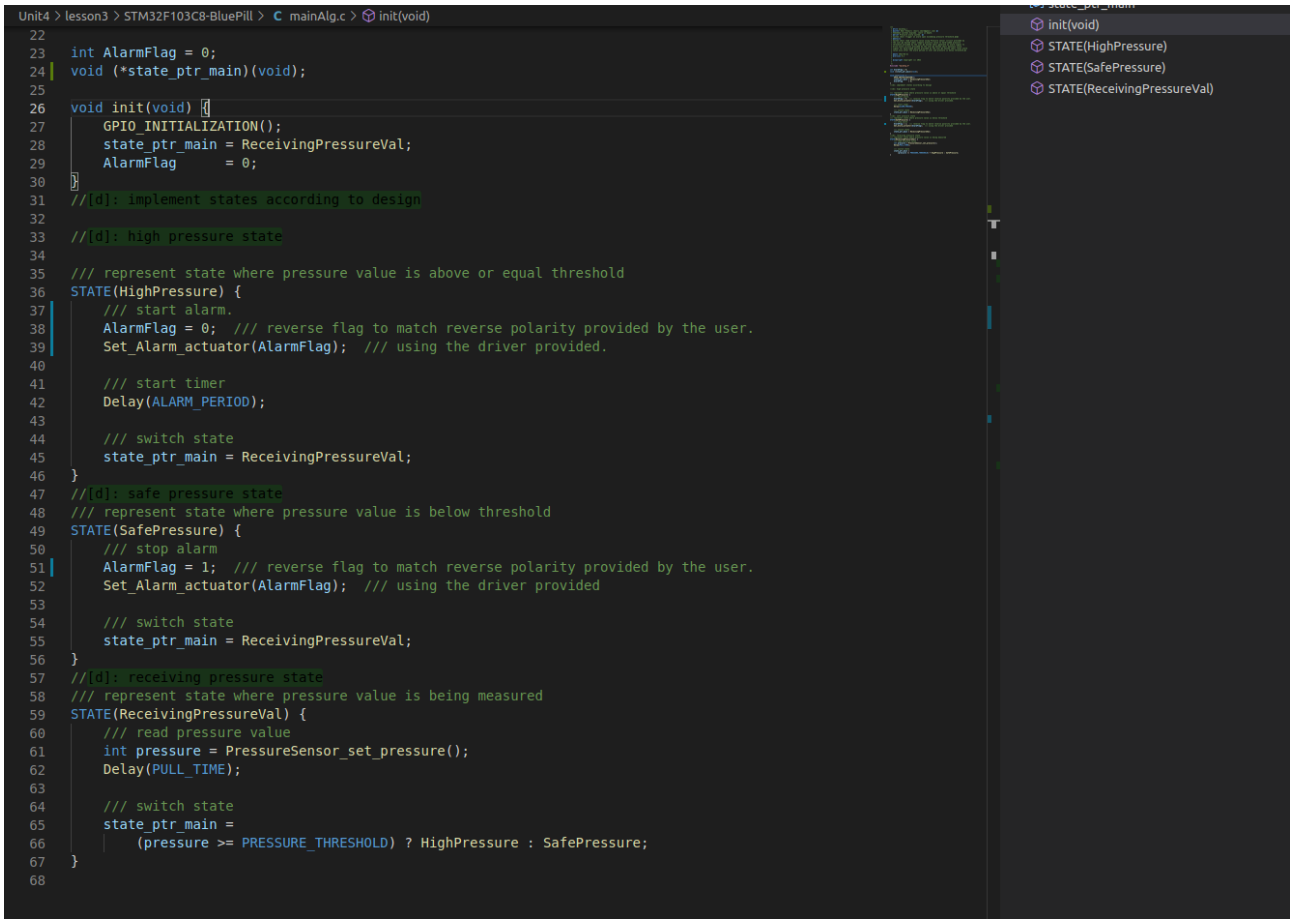


Figure 6: Main Algorithm states

Also, follow the link for state machine simulation

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

6 Source Code




```c
int AlarmFlag = 0;
void (*state_ptr_main)(void);

void init(void) {
    GPIO_INITIALIZATION();
    state_ptr_main = ReceivingPressureVal;
    AlarmFlag      = 0;
}
//[d]: implement states according to design

//[d]: high pressure state

/// represent state where pressure value is above or equal threshold
STATE(HighPressure) {
    /// start alarm.
    AlarmFlag = 0;  /// reverse flag to match reverse polarity provided by the user.
    Set_Alarm_actuator(AlarmFlag);  /// using the driver provided.

    /// start timer
    Delay(ALARM_PERIOD);

    /// switch state
    state_ptr_main = ReceivingPressureVal;
}
//[d]: safe pressure state
/// represent state where pressure value is below threshold
STATE(SafePressure) {
    /// stop alarm
    AlarmFlag = 1;  /// reverse flag to match reverse polarity provided by the user.
    Set_Alarm_actuator(AlarmFlag);  /// using the driver provided

    /// switch state
    state_ptr_main = ReceivingPressureVal;
}
//[d]: receiving pressure state
/// represent state where pressure value is being measured
STATE(ReceivingPressureVal) {
    /// read pressure value
    int pressure = PressureSensor_set_pressure();
    Delay(PULL_TIME);

    /// switch state
    state_ptr_main =
        (pressure >= PRESSURE_THRESHOLD) ? HighPressure : SafePressure;
}
```

Figure 7: Main Algorithm .c file

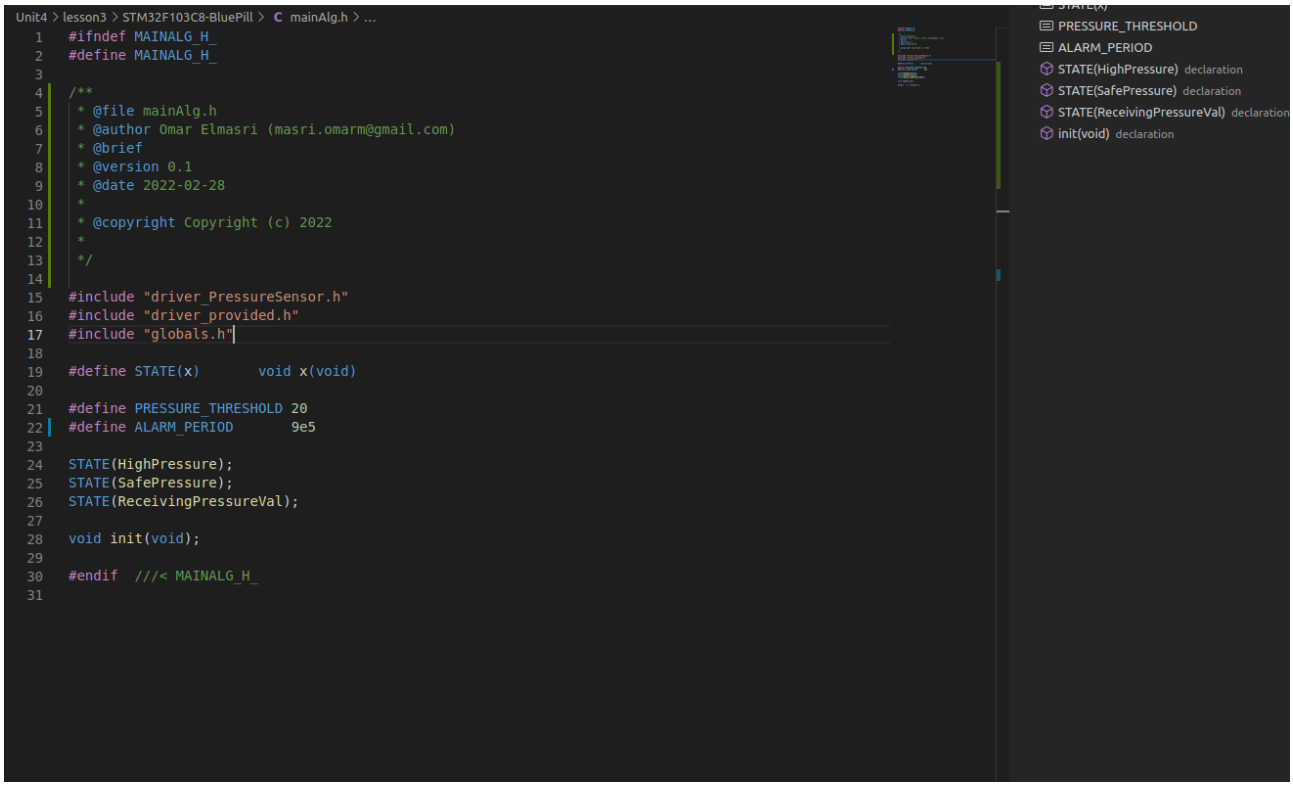


Figure 8: Main Algorithm .h file

7 Simulation

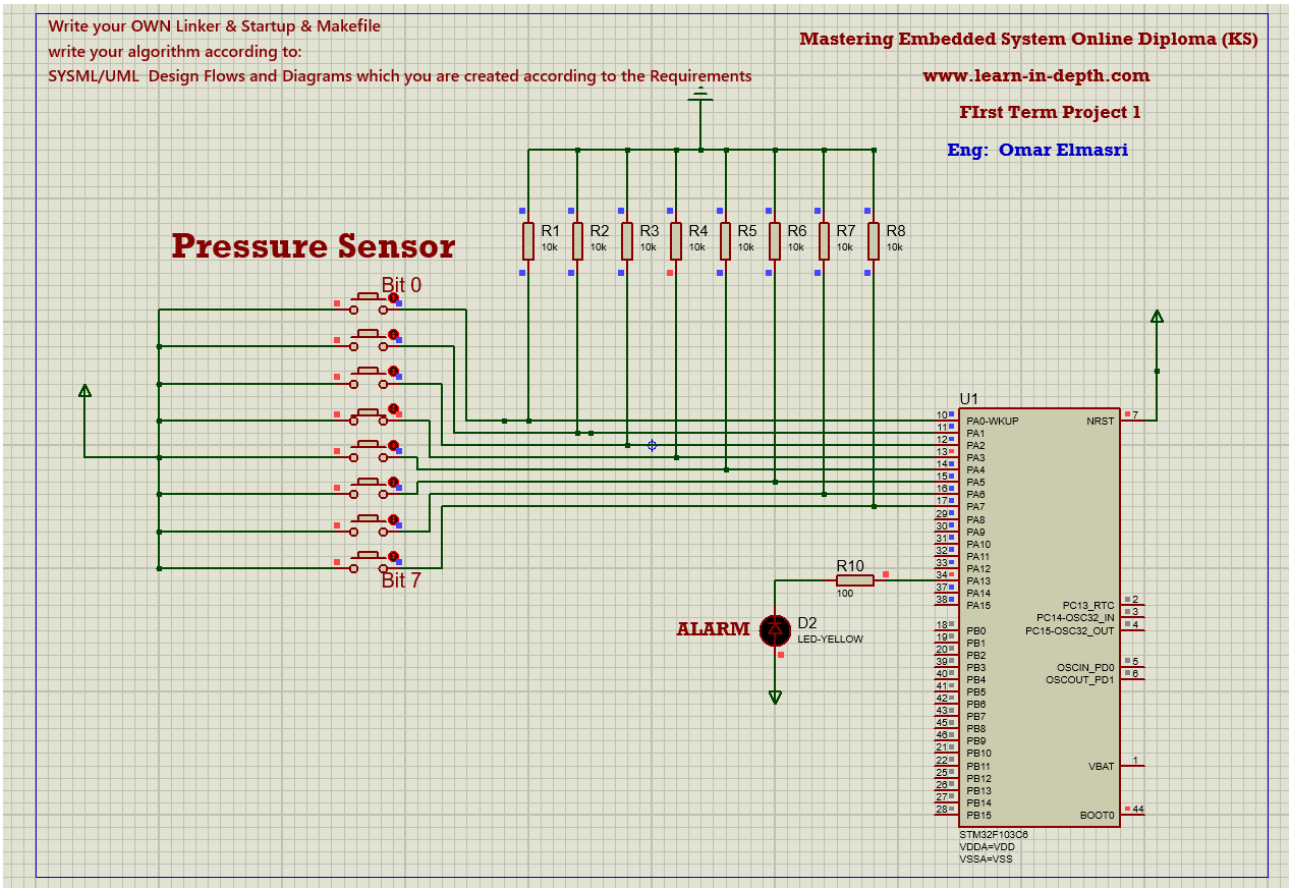


Figure 9: pressure below threshold of 20, LED is off

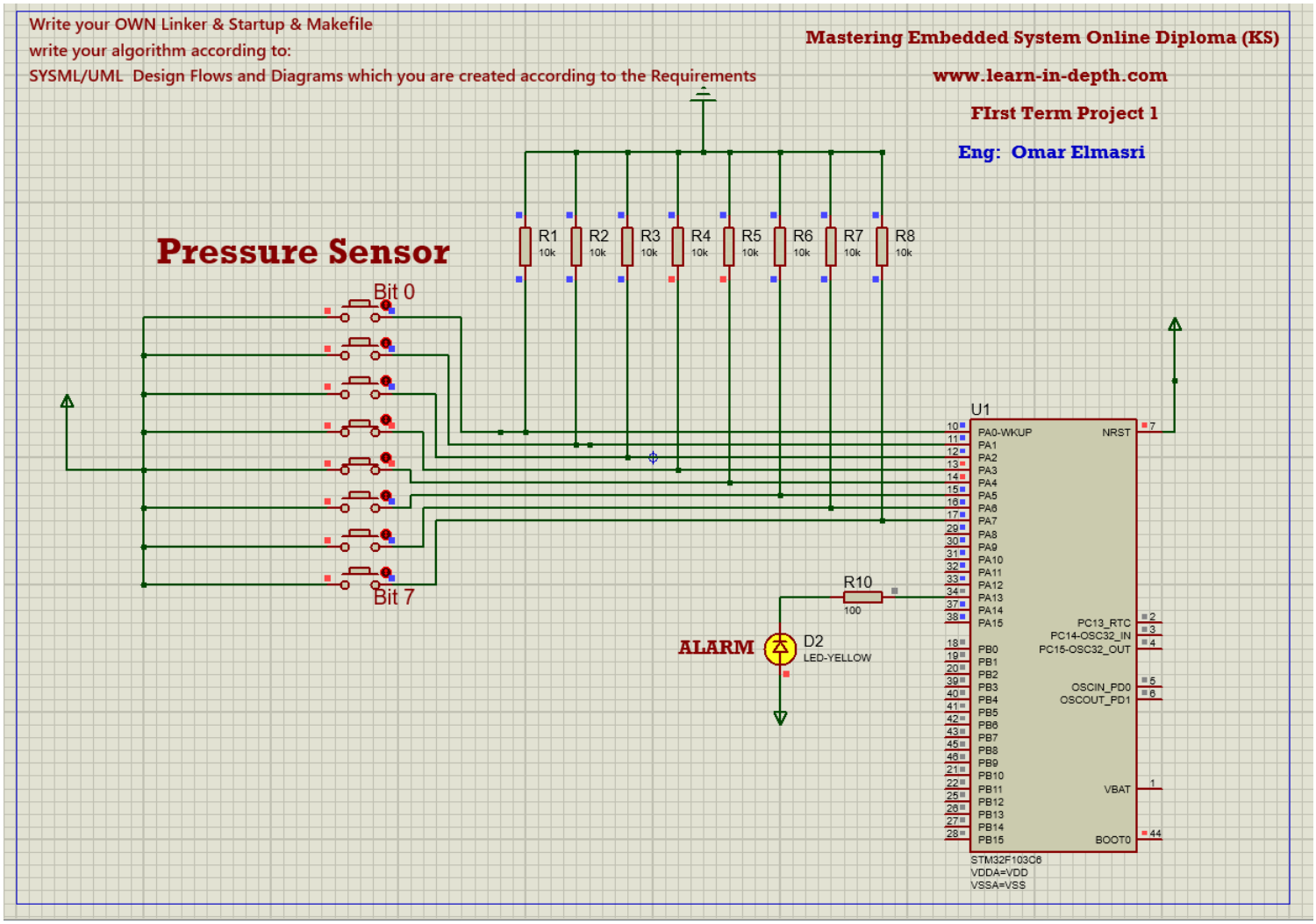


Figure 10: Pressure is 24 above threshold of 20, LED is on

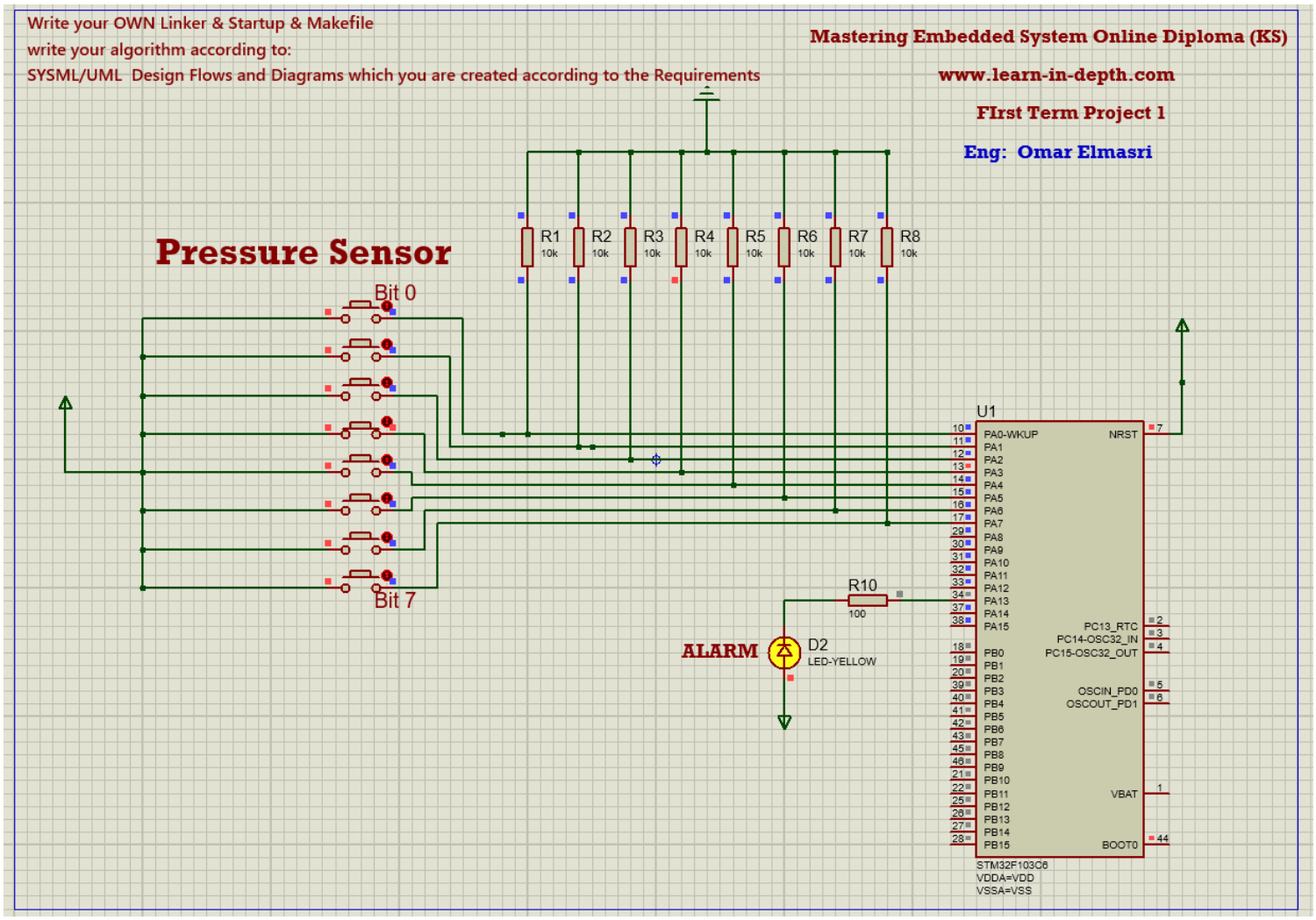


Figure 11: Pressure is below threshold, yet, LED is on due to delay