

Embedded C Assignment

Austen Thomas

Student Number: 2418505

Electrical & Electronics Engineering

L4- Embedded C

ACDA4013

Word count – 1725

Figure 1 Illustrates PIC16F877A microcontroller	4
Figure 2 Illustrates Push Button and resistor.....	5
Figure 3 Illustrates Light Emitting Diode (LED)	5
Figure 4 Illustrates Liquid Crystal Display (LCD).....	6
Figure 5 Illustrates Crystal Oscillator and Capacitor	6
Figure 6 Illustrates 5 V DC Motor.....	7
Figure 7 Illustrates Pins OSC1, OSC2, MCLR.....	8
Figure 8 Illustrates Switch Button Circuit and Connected Pins.....	9
Figure 9 Illustrates LEDs and pin connections	9
Figure 10 Illustrates LCD components and wiring connections.....	10
Figure 11 Illustrates 5 V DC motor and wiring connections	11
Figure 12 Illustrates Virtual terminal wiring and virtual terminal messages.....	11
Figure 13 Illustrates Flow Chart	13
Figure 14 Illustrates system ready to be initialised	14
Figure 15 Illustrates Engine start system	15
Figure 16 Illustrates Coolant OK system sequence.....	15
Figure 17 Illustrates Fan activated and motor running	16
Figure 18 Illustrates System shutdown with LCD displays "Alert!" and "Shutdown done"	17

Table of Contents

Introduction	3
Background Research.....	3
Circuit Diagrams.....	8
Flowchart	12
Screenshots.....	14
Details of Testing.....	18
Conclusion.....	19
References	20
Appendix	21

Introduction

This project focuses on the design and simulation of an automotive cooling system using a PIC16F877A microcontroller. The aim of the system is to replicate the logical sequence of events that would occur in a real vehicle cooling process, including engine start, coolant monitoring, fan activation, and system shutdown. The system uses three push-button switches as inputs to simulate real-world sensors, and provides feedback through LEDs, an LCD display, and a 5 v DC motor. Each stage of the system is triggered in a specific order, ensuring that the process follows a controlled sequence. The LCD is used to display real-time system messages, while LEDs provide a visual indication of each stage. A motor is included to simulate the operation of a radiator cooling fan. This project demonstrates how embedded systems can be used to control and monitor processes in real-time applications such as automotive systems and control systems.

Background Research

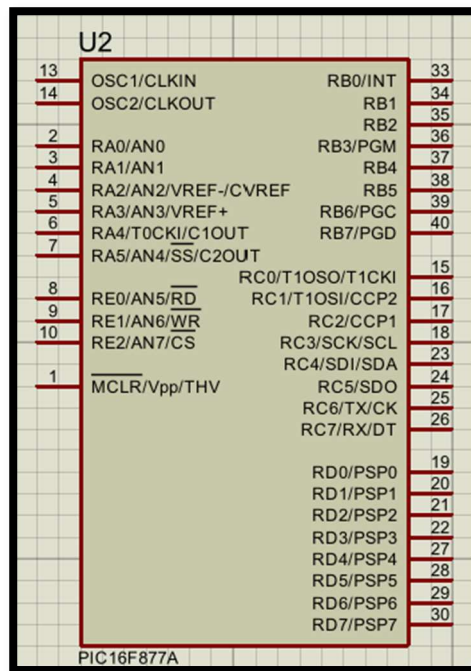
The problem domain of this project is the simulation of an automotive engine cooling system using an embedded microcontroller. In modern vehicles, embedded systems are essential for monitoring engine conditions and controlling auxiliary systems such as cooling fans to prevent overheating. These systems operate in real time, responding to sensor inputs and executing control actions to maintain safe and efficient operation [1].

Embedded systems are specialised computing systems designed to perform dedicated functions within a larger system. Embedded systems are optimised for reliability, low power consumption and deterministic performance [2]. In automotive applications, embedded systems are widely used for engine management, temperature monitoring and safety critical control processes [1].

The first component used is the PIC16F877A microcontroller which controls the system which features a central processing unit, memory and input/output ports

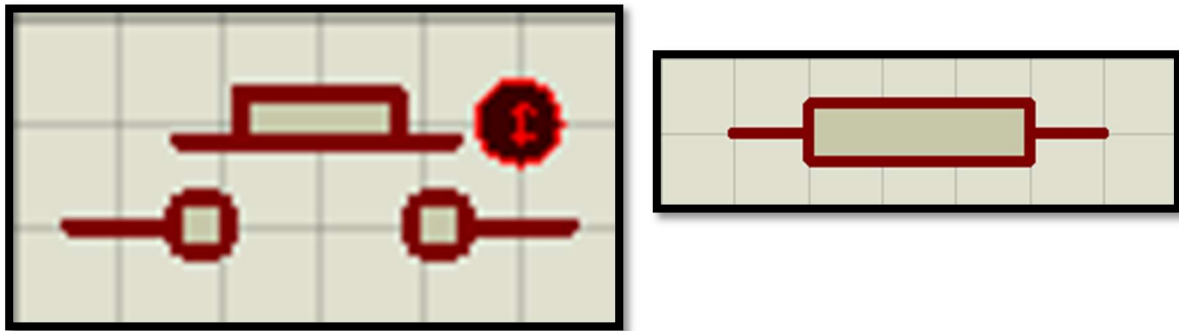
on a single chip. Microcontrollers are commonly used in embedded systems due to their ability to interface with both input devices and output components. The microcontroller processes signals from input switches and controls outputs such as LEDs, an LCD display and a motor [3].

Figure 1 Illustrates PIC16F877A microcontroller



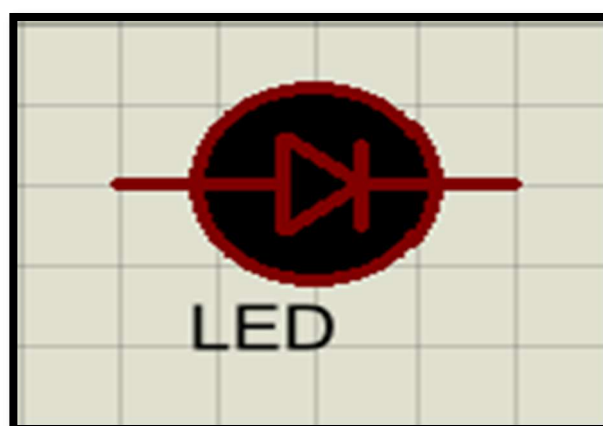
The next component used is push button switches, which simulate real-world sensors. These include engine start trigger, coolant sensor trigger and fan activation trigger. To ensure stable operation, resistors are used to prevent floating inputs, which can cause unpredictable behaviour in digital systems [4].

Figure 2 Illustrates Push Button and resistor



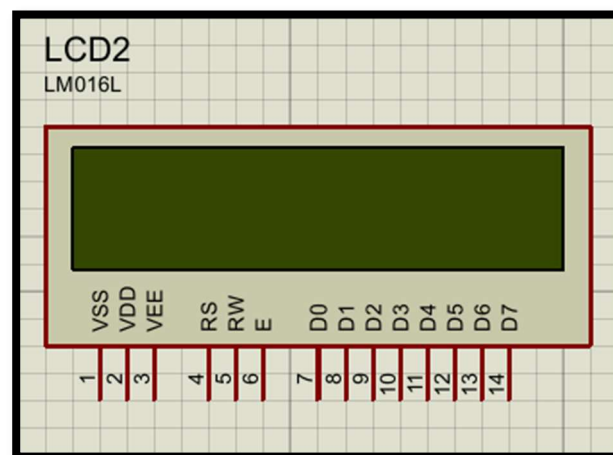
Visual feedback is provided using Light Emitting Diodes (LEDs, which indicate different stages of the system operation. LEDs are widely used in embedded systems due to their low power consumption, fast response time, and ease of interfacing with microcontrollers [5].

Figure 3 Illustrates Light Emitting Diode (LED)



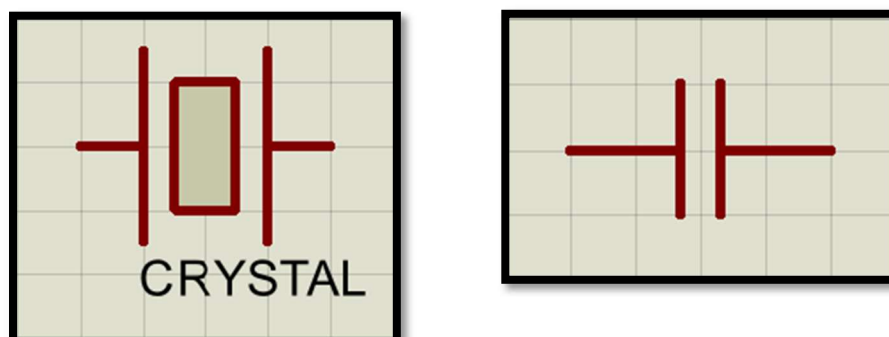
Another component used is a liquid crystal display (LCD) module is used to provide real time textual feedback to the user. LCDs are commonly used in embedded applications because they are energy efficient and capable of displaying alphanumeric characters clearly. In this system, the LCD operates in 4-bit mode, reducing the number of required data lines while maintaining effective communication with the microcontroller [6] .

Figure 4 Illustrates Liquid Crystal Display (LCD)



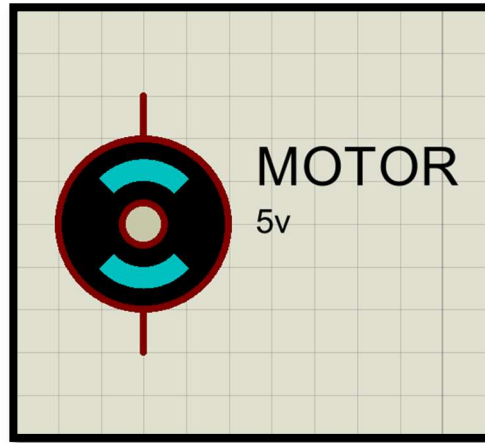
A quartz crystal oscillator is used to provide a stable clock signal to the microcontroller, ensuring accurate timing and reliable execution of instructions [7]. Capacitors are used alongside the crystal to stabilise oscillations [8].

Figure 5 Illustrates Crystal Oscillator and Capacitor



Finally, a DC motor is used to simulate the cooling fan, demonstrating how embedded systems can control physical devices based on input conditions.

Figure 6 Illustrates 5 V DC Motor

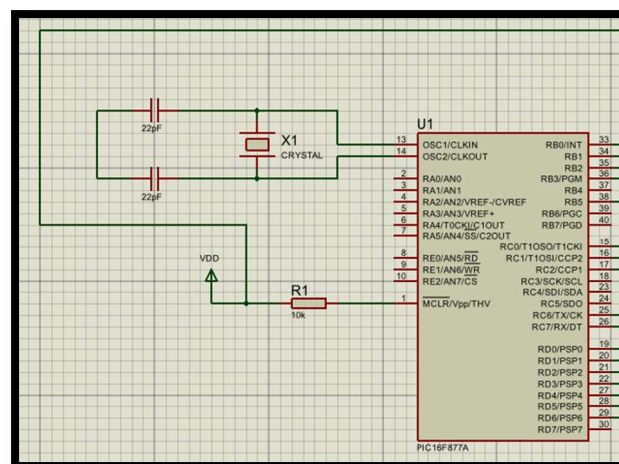


Circuit Diagrams

The circuit was designed and simulated using proteus software, implementing a complete embedded control system based on the PIC16F877A microcontroller. The design integrates input buttons, LED indicators, LCD display module, and a motor to simulate an automotive cooling system.

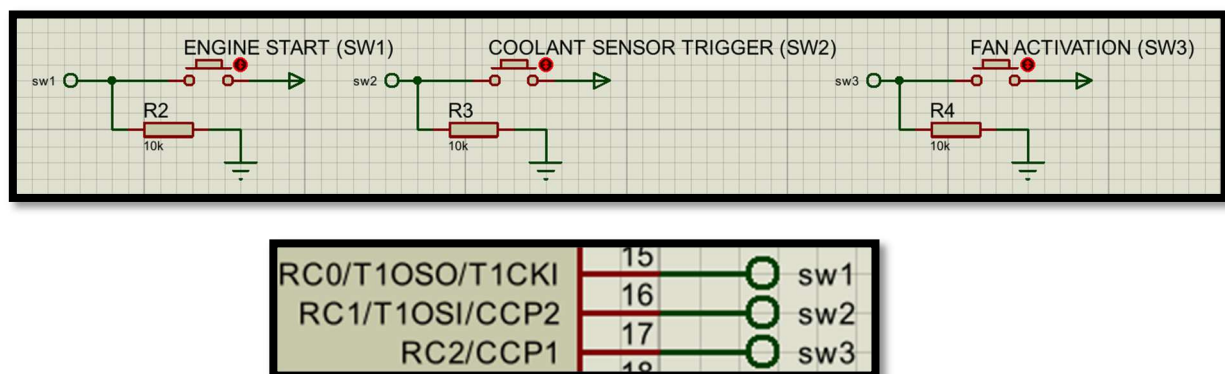
At the centre of the circuit is the PIC16F877A microcontroller, operating at a clock frequency of 20 MHz. A quartz crystal oscillator is connected between the OSC1 and OSC2 pins, with two 22 pF capacitors connected to ground to stabilise the oscillation. This provides a reliable clock signal required for accurate timing and execution of the program. The MCLR pin is connected to the supply voltage through a 10 K Ω resistor to ensure the microcontroller operates correctly and avoids any unintended problems.

Figure 7 Illustrates Pins OSC1, OSC2, MCLR



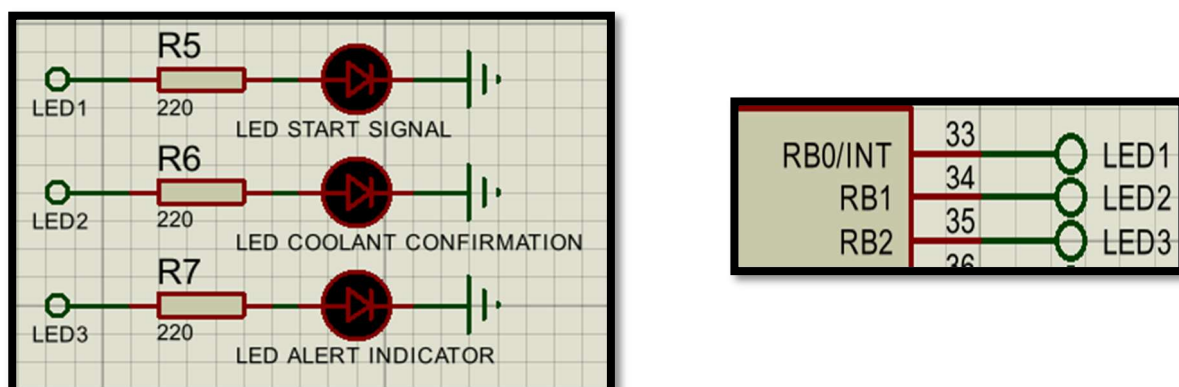
The next section of the circuit is the input section which consists of three push button switches connected to pins RC0, RC1 and RC2 of the PORTC. These button switches represent the engine start, coolant sensor trigger and fan activation trigger. Each switch is connected with a 10 K Ω resistor to ground. This ensures that when the buttons are not pressed, the input pins remain at a low input level, preventing floating inputs which could lead to unpredictable circuit behaviour [9].

Figure 8 Illustrates Switch Button Circuit and Connected Pins



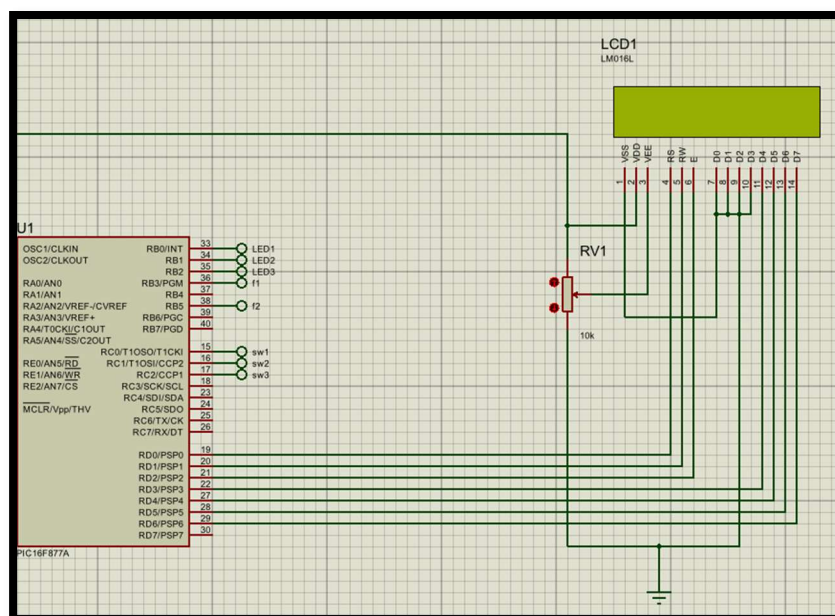
The following section of the circuit includes LEDs connected to pins RB0, RB1, and RB2 of PORTB. Each LED is connected in series with a 220 Ω resistor to limit current and protect the LEDs from damage. These LEDs provide visual feedback corresponding to system states, including LED engine start signal, LED coolant confirmation, LED alert indicator.

Figure 9 Illustrates LEDs and pin connections



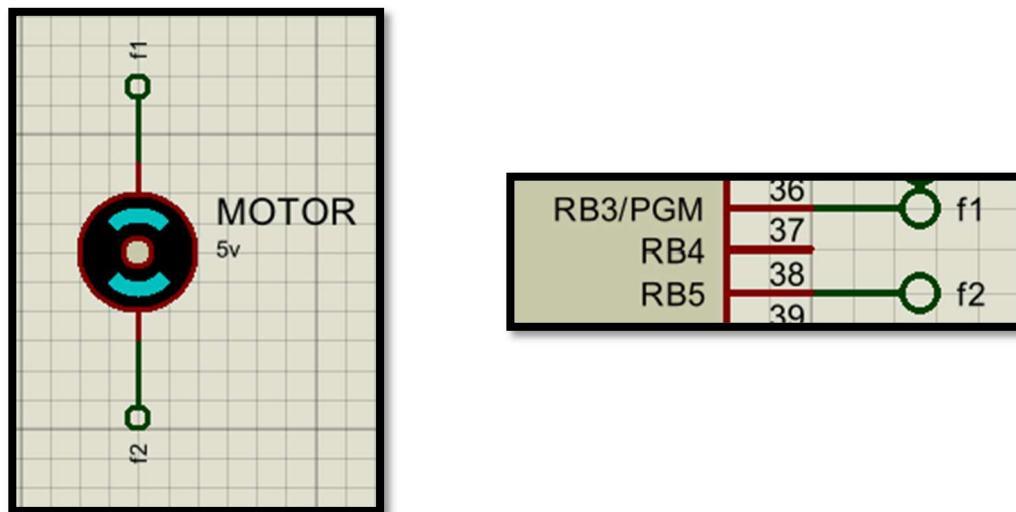
A 16 x 2 LM016L LCD module is interfaced with the microcontroller using PORTD in 4-bit mode. The data lines on the LCD D4, D5, D6 and D7 are connected to pins RD3, RD4, RD5 and RD6 on the microcontroller, while the control pins RS, RW, and E are connected to RD0, RD1 and RD2. A 10 K Ω potentiometer is connected to the VEE pin of the LCD to adjust contrast, ensuring that displayed care clearly visible. The use of 4-bit mode reduces the number of I/O pins required, allowing more efficient utilisation of the microcontroller.

Figure 10 Illustrates LCD components and wiring connections



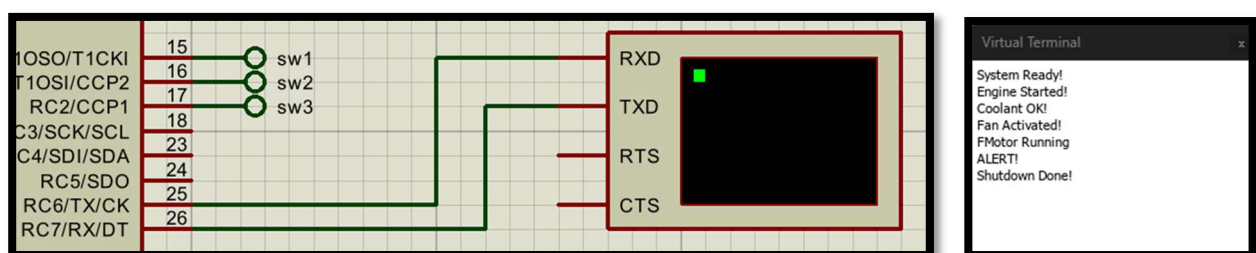
Furthermore a 5 V DC motor is included in the circuit to simulate the cooling fan. The motor is controlling through output pins RB3 and RB5, which determine its activation state. When the required conditions are met, the microcontroller outputs signals to drive the motor, representing the operation of the cooling system.

Figure 11 Illustrates 5 V DC motor and wiring connections



Additionally, UART communication is added using pins RC6 and RC7 which are connected to pins RXD and TXD on a virtual terminal in proteus. This allows systems alerts and messages to be transmitted and monitored in real time.

Figure 12 Illustrates Virtual terminal wiring and virtual terminal messages



Overall, the circuit demonstrates a complete embedded system, integrating inputs, processing and outputs to simulate a real-world automotive cooling application.

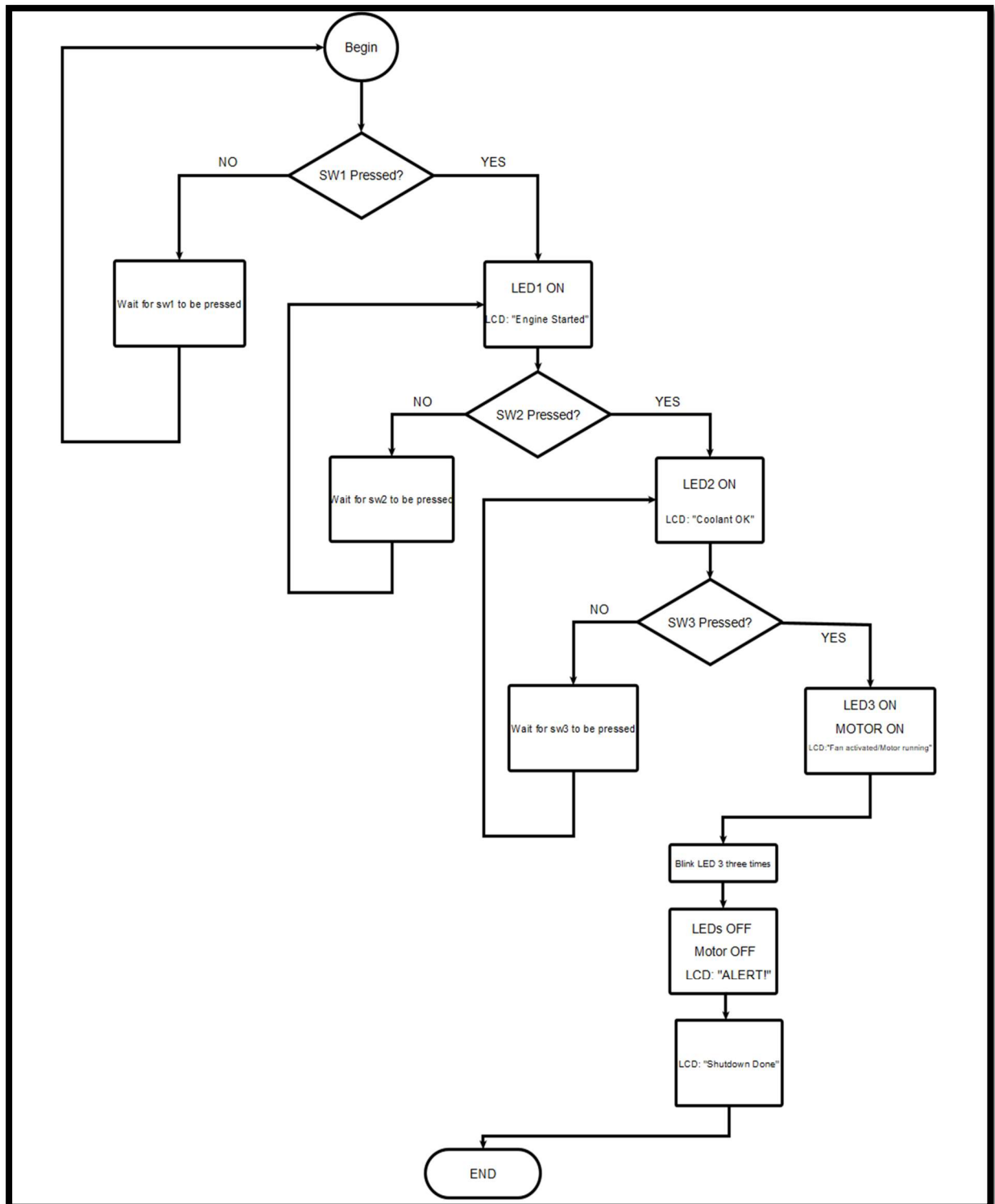
Flowchart

The flowchart represents the logical operation and control sequence of an automotive cooling system. It provides a visual overview of how the system respond to input switches and controls outputs such as LEDs, the motor, and the LCD display. The process beings with system initialisation, where all outputs are set to an off state and the LCD displays “System ready”.

The system then operates using a sequential decision- making structure based on three push-button inputs (SW1, SW2 and SW3). Each stage is dependent on the correct order of activation. The flowchart uses decision nodes to check whether each switch has been pressed. If a switch is not pressed, the system remains in a continuous loop at that stage, effectively waiting for the required input before progressing. This ensures that the sequence cannot advance prematurely.

Once a valid input is detected, the system performs the corresponding actions. These include activating LEDs to indicate system status and updating the LCD with appropriate messages such as “Engine Started”, “Coolant OK”, and “Fan Activated / Motor Running”. Upon completion of the sequence, the system triggers an alert phase where the final LED blinks, followed by all outputs being turned off. The LCD then displays “ALERT!” and finally “Shutdown Done”, completing the operation cycle.

Figure 13 Illustrates Flow Chart



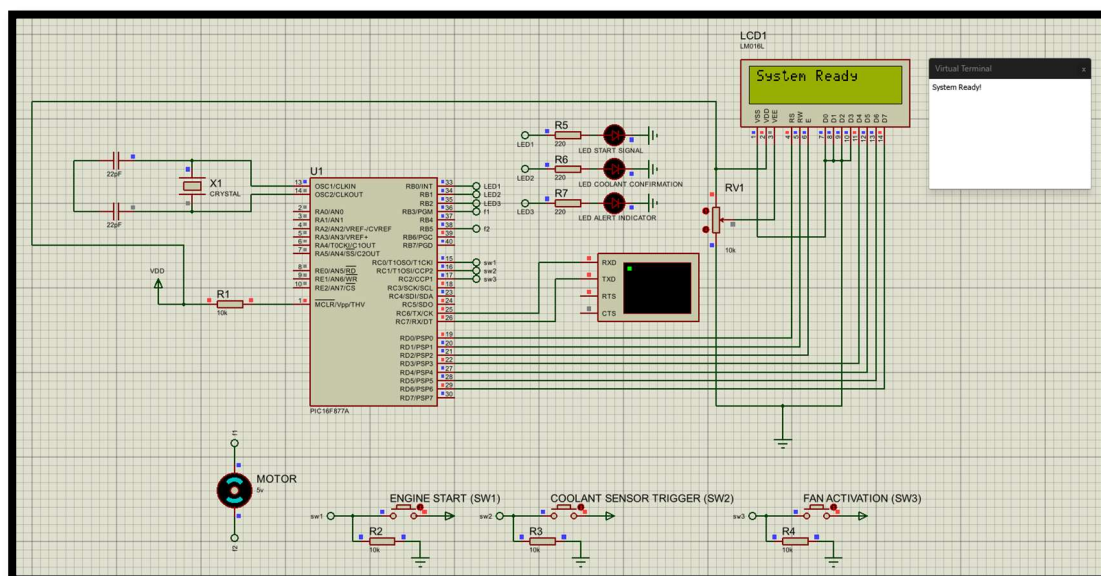
Screenshots

This section presents a series of screenshots taken from the proteus simulation to demonstrate the correct operation of the automotive cooling system at each key stage.

1. System initialisation

The system powers on with all LEDs turned off and the motor is inactive. The LCD displays “System Ready”, confirming that the microcontroller has been correctly initialised and is awaiting user input.

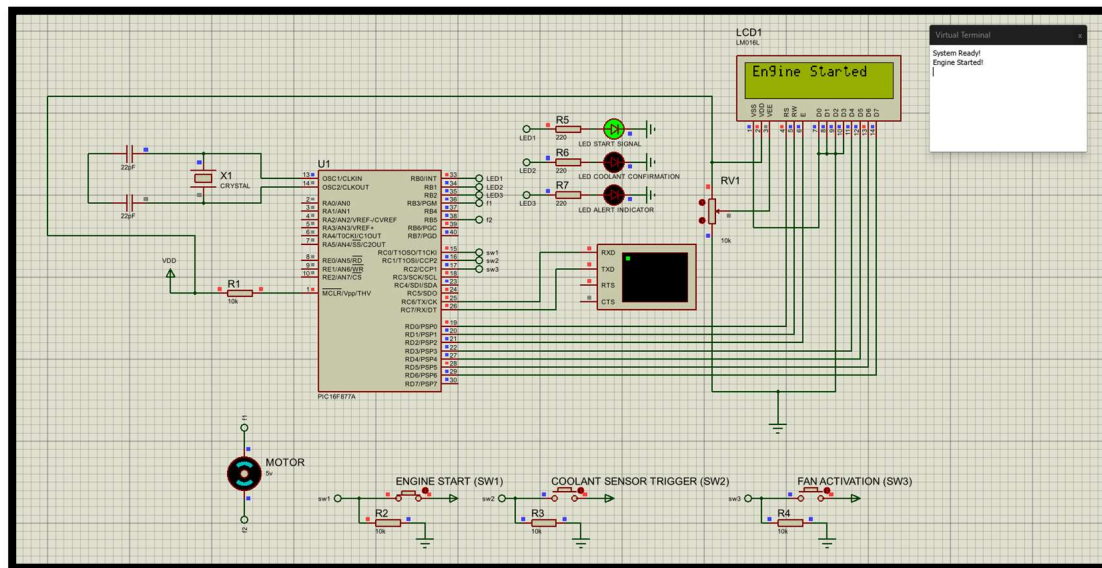
Figure 14 Illustrates system ready to be initialised



2. Engine Start (SW1 pressed)

When SW1 is pressed, LED 1 is activated to indicate the engine start signal. The LCD updates to display “Engine Started”, confirming that the input has been correctly pressed and processed.

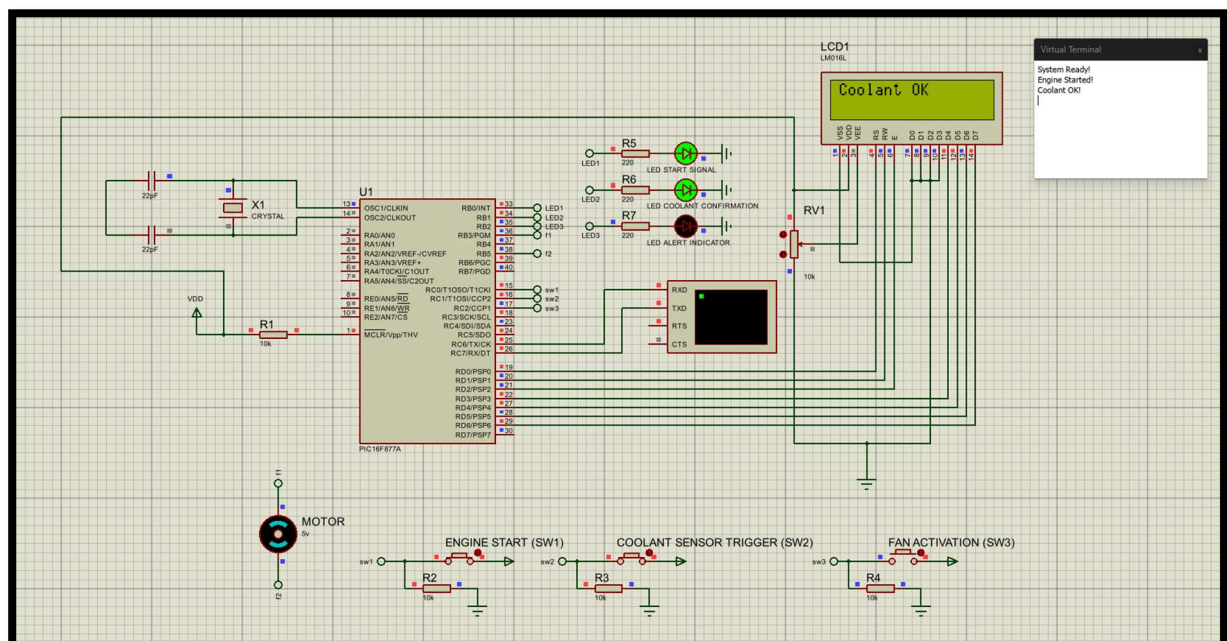
Figure 15 Illustrates Engine start system



3. Coolant confirmation (SW2 pressed)

After SW2 is pressed, LED2 turns on to represent coolant confirmation. The LCD displays “Coolant OK”, demonstrating correct progression through the system sequence.

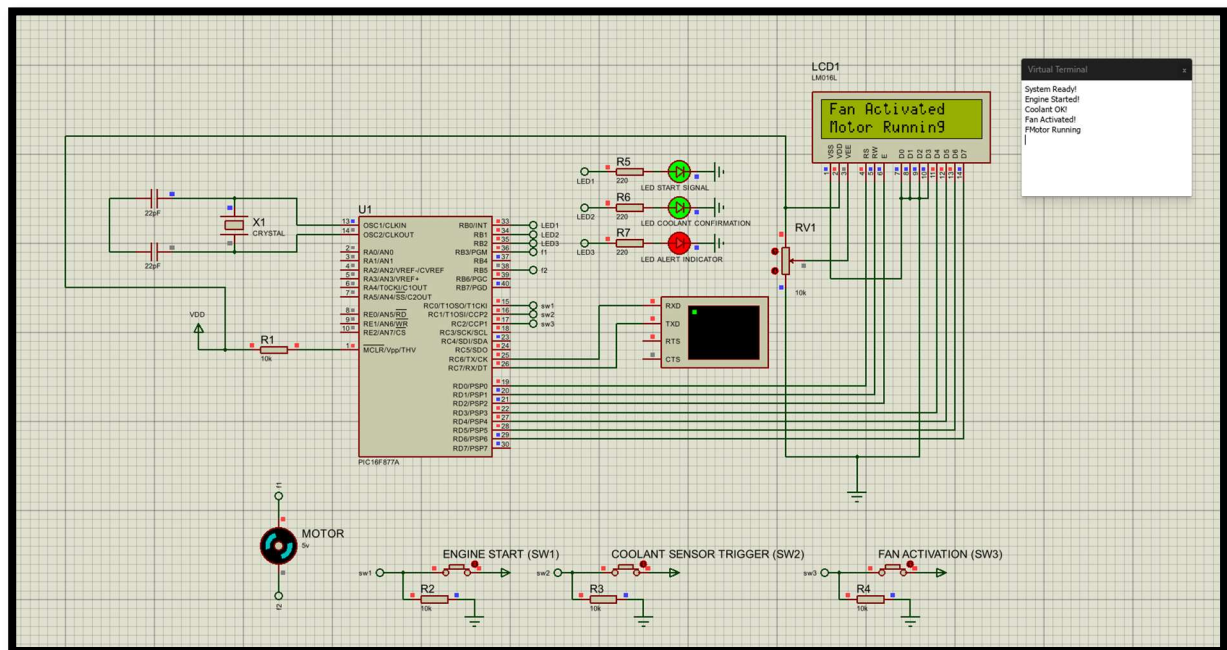
Figure 16 Illustrates Coolant OK system sequence



4. Fan activation and motor running (SW3)

When SW3 is pressed, LED3 is activated and the motor begins to run. THE LCD displays “Fan Activated” and “Motor Running”, confirming that both visual and mechanical outputs are functioning correctly.

Figure 17 Illustrates Fan activated and motor running



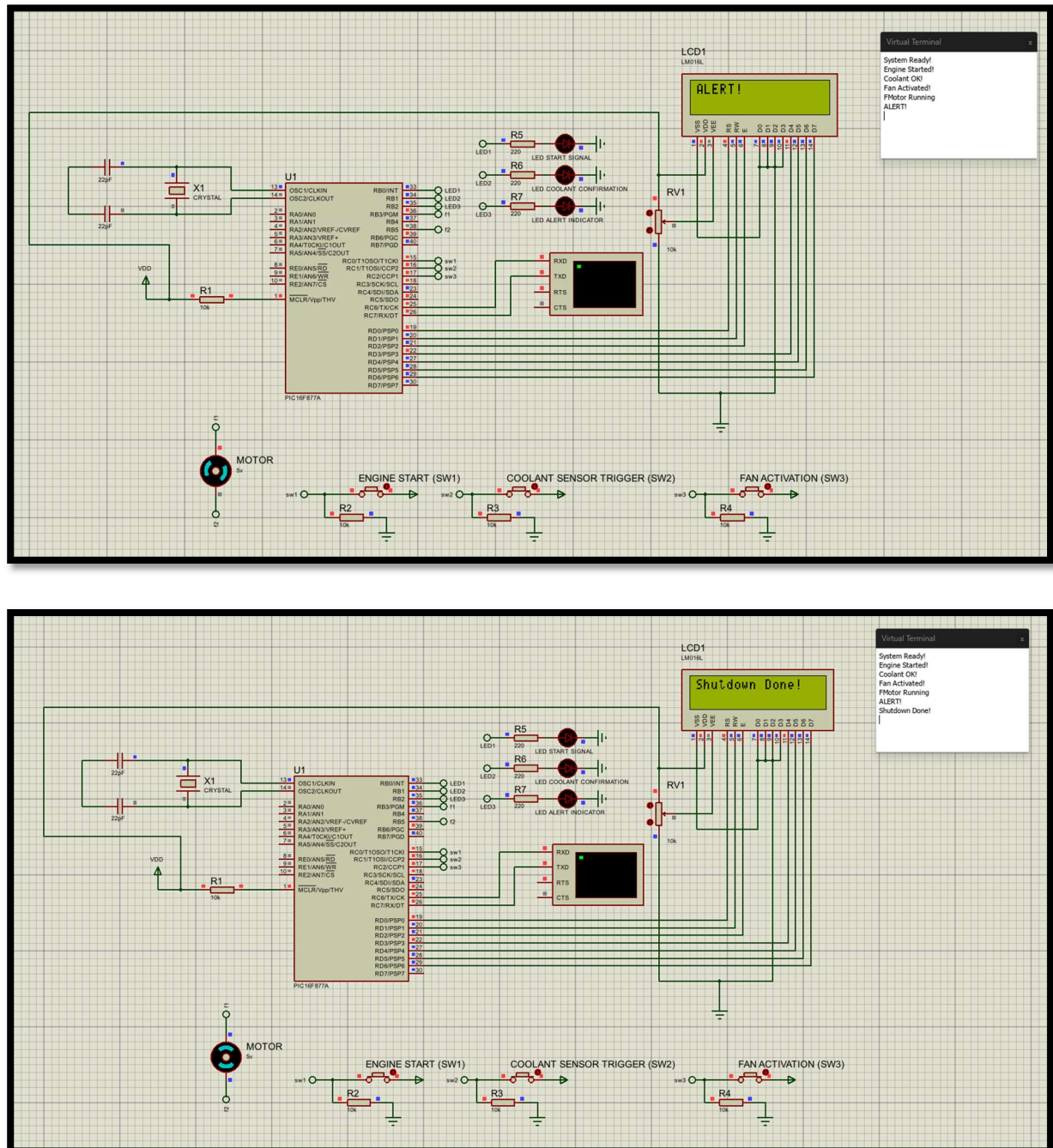
5. Alert Indication (LED Blinking)

Following fan activation, LED3 three times at 100ms intervals. This behaviour simulates an alert condition and demonstrates the use of timing delays withing the program.

6. System Shutdown

After the alert sequence, all LEDs are turned off and the motor stops. The LCD displays “Alert!” followed by “Shutdown Done”, indicating that the system has completed its operational cycle.

Figure 18 Illustrates System shutdown with LCD displays "Alert!" and "Shutdown done"



These simulation results verify that the system responds correctly to each input and follows the intended sequence program

Details of Testing

The system was tested in proteus to verify correct operation at each stage of the control sequence.

Test (Initial State)

When no switches are pressed, the expected behaviour was that all LEDs remain off, the motor is inactive, and the LCD displays “System ready”. The simulation confirmed this behaviour.

Test (Engine Start)

When SW1 was pressed, LED1 was expected to turn on and the LCD to display “Engine Started”. The observed output, matched expectations.

Test (Coolant Confirmation)

Pressing SW2 was expected to activate LED2 and update the LCD to “Coolant OK”. The simulation produced the correct result, showing that the system progressed to the next stage only after SW1 was triggered.

Test (Fan Activation)

When SW3 was pressed, LED3 and the motor were expected to turn on, with the LCD displaying “Fan Activated” and “Motor Running”. The results matched the expected behaviour.

Test (Alert and Shutdown)

The LED3 blinking sequence occurred three times as programmed. After this, all LEDs turned off, the motor stopped, and the LCD displayed “Alert!” followed by “shutdown done”, confirming correct shutdown behaviour.

Overall, the system performs as intended.

Conclusion

This project successfully demonstrated the design and simulation of an automotive cooling system using a PIC 16F877A microcontroller. The system operated as intended, with each stage of the sequence correctly triggered by the corresponding switch inputs. Outputs including LEDs, the LCD display, and the motor responded accurately, providing clear visual and functional feedback.

The use of sequential logic ensured that each stage could only be activated in the correct order, reflecting real-world behaviour. Simulation results confirmed reliable system performance, with all expected outputs matching actual behaviour during testing.

References

- [1] Amoria Bond, "What role do embedded systems play in automotive development?," Amoria Bond, 12 September 2024. [Online]. Available: <https://www.amoriabond.com/insights/articles/what-role-do-embedded-systems-play-in-automotive-development-/>. [Accessed 9 April 2026].
- [2] RS components, "Understanding Embedded systems," RS components, 19 May 2025. [Online]. Available: <https://uk.rs-online.com/web/content/discovery/ideas-and-advice/embedded-systems-guide>. [Accessed 9 April 2026].
- [3] Florius, "Introduction to the PIC16F877A microcontroller," Florisera, 15 August 2025. [Online]. Available: <https://florisera.com/introduction-to-the-pic17f877a-microcontroller/>. [Accessed 9 April 2026].
- [4] Electronics Tutorials, "Pull-up Resistors," Electronics Tutorials, [Online]. Available: <https://www.electronics-tutorials.ws/logic/pull-up-resistor.html>. [Accessed 9 April 2026].
- [5] P. Peddeti, "Types of LED lights & their applications in embedded systems," embedkari, 29 March 2025. [Online]. Available: <https://embedkari.org/2025/03/29/types-of-led-lights-their-applications-in-embedded-systems/>. [Accessed 9 April 2026].
- [6] ScienceDirect, "Liquid Crystal Display," ScienceDirect, [Online]. Available: <https://www.sciencedirect.com/topics/computer-science/liquid-crystal-display>. [Accessed 9 April 2026].
- [7] A. E. M. Qin Zhuang, "Crystal Oscillators: Simple, Low-Cost and Highly Accurate Clock Sources," Diodes Analog & Power Solutions, 2019. [Online]. Available: <https://www.diodes.com/design/support/perspective/crystal-oscillators-simple-low-cost-and-highly-accurate-clock-sources>. [Accessed 9 April 2026].
- [8] YXC, "3 Benefits of Connecting 2 Capacitors to a Crystal Oscillator," YXC, [Online]. Available: <https://www.yxcxtal.com/news/yxc-knowledge-popularization---3-benefits-of-connecting-2-capacitors-to-a-crystal-oscillator.html>. [Accessed 9 April 2026].
- [9] ThinkRobotics, "Understanding Pull-Up and Pull-Down Resistors in Electronics," ThinkRobotics, 27 November 2024. [Online]. Available: https://thinkrobotics.com/blogs/learn/understanding-pull-up-and-pull-down-resistors-in-electronics?srsId=AfmBOoqt9GH-T__2KMHlkv8pvF2dGfIcoNPqs11S7ydtzOsm5zjgOsE. [Accessed 9 April 2026].

Appendix

Project Wizard - C:\Users\s2418505\Documents\Elec Eng Degree Folder\L4\Embedded C\Chaser sequence\Project Backups\main.ccs... X

File Help

Options Code

Peripherals

Analog

Communications

SPI

Drivers

Header Files

High/Low Voltage

Interrupts

I/O Pins

Timers 0-2

Timers 3-up

LCD (Internal)

LCD (External)

Capacitive Touch

RTCC

CCP/Vref

General

Device

Family: PIC16

Device: PIC16F877A

Debug Code ☐

Fixed Compiler Version: None

Clock

Oscillator Type: Crystal

Crystal Clock Speed: 20,000,000 Hz 5 MIPS

CPU Clock Speed: 0 Hz

Use USB Low Speed ☐

Use USB Full Speed ☐

Clock Out ☐

WDT

☐ Enabled

Check any of the following to restart WDT during calls to:

☐ getc() and fgetc()

☐ i2c_read()

☐ Delay Functions

Reset: 18.0 ms

WDT Reset

☒ 18 ms

☐ 36 ms

☐ 72 ms

☐ 144 ms

☐ 288 ms

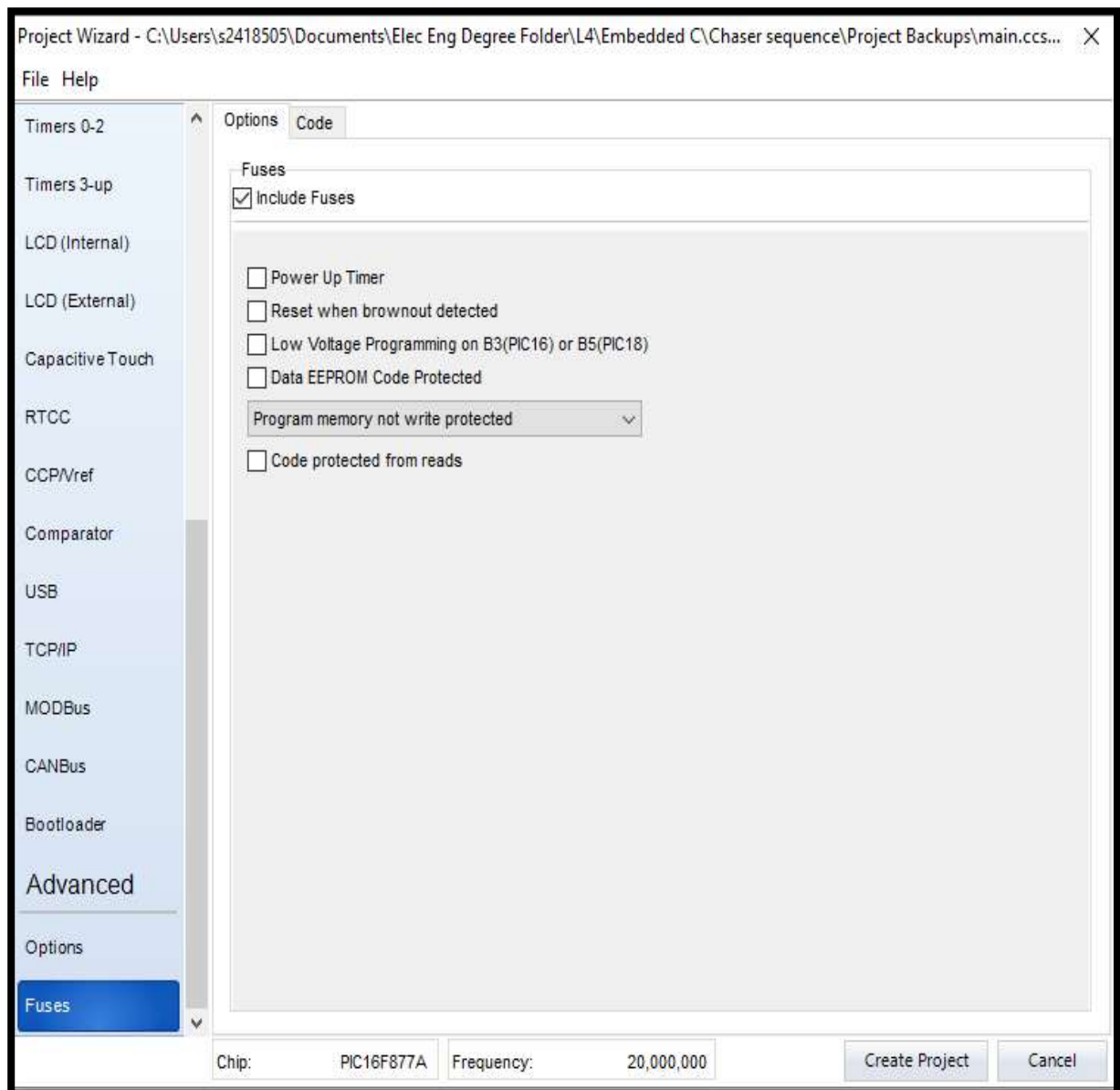
☐ 576 ms

☐ 1152 ms

☐ 2304 ms

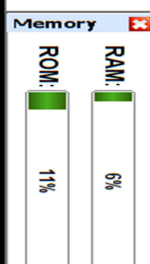
Chip: PIC16F877A Frequency: 20,000,000

Create Project Cancel





Compiling E:\EEEP Degree Apprenticeship\Level 4\Embedded C\EEEP\Embedded C Assignment\Embedded C assignment task on 24-Mar-26 at 17:17
 Memory usage: ROM=11% RAM=2% - 6%
 0 Errors, 0 Warnings.
 Build Successful



```

1  #include <16F877A.h>
2  #fuses HS, NOWDT, NOLVP
3  #use delay(clock=20000000)
4  #use rs232(baud=9600, xmit=PIN_C6, rcv= PIN_C7,bits=8, parity=N, stop=1)
5
6  // ----- LCD (4-bit mode) -----
7  #define LCD_RS_PIN PIN_D0
8  #define LCD_ENABLE_PIN PIN_D2
9  #define LCD_RW_PIN PIN_D1
10
11
12  #define LCD_DATA4 PIN_D3
13  #define LCD_DATA5 PIN_D4
14  #define LCD_DATA6 PIN_D5
15  #define LCD_DATA7 PIN_D6
16
17  #include <lcd.c>
18
19
20
21  // ----- Define pins -----
22  #define LED1 PIN_B0
23  #define LED2 PIN_B1
24  #define LED3 PIN_B2
25
26  #define SW1 PIN_C0
27  #define SW2 PIN_C1
28  #define SW3 PIN_C2
29
30  //----- Define MOTOR -----
31  #define f1 PIN_B3
32  #define f2 PIN_B5
33
34  void main()
35  {
36      // ----- Set ports -----
37      set_tris_b(0x00); // LEDs output
38      set_tris_c(0x07); // switches input
39      set_tris_d(0x00); // LCD output
40
41      lcd_init();
42
43      // ----- Initial state -----
44      output_low(LED1);
45      output_low(LED2);
46      output_low(LED3);
47      output_low(f1);
48      output_low(f2);
49
50

```

```
49
50
51
52     lcd_putc("\fSystem Ready");
53     printf("System Ready!\r\n");
54     delay_ms(1500);
55
56
57     // ----- STEP 1: ENGINE START -----
58
59
60     while(!input(SW1));
61         delay_ms(50);
62         output_high(LED1);
63         lcd_putc("\fEngine Started");
64         printf("Engine Started!\r\n");
65         delay_ms(1500);
66
67     // ----- STEP 2: COOLANT -----
68
69
70     while(!input(SW2));
71         delay_ms(50);
72         output_high(LED2);
73         lcd_putc("\fCoolant OK");
74         printf("Coolant OK!\r\n");
75         delay_ms(1500);
76
77     // ----- STEP 3: FAN -----
78
79     while(!input(SW3));
80         delay_ms(50);
81         output_high(LED3);
82         lcd_putc("\fFan Activated\nMotor Running");
83         printf("Fan Activated!\r\n");
84         printf("FMotor Running\r\n");
85         output_high(f1);
86         output_low(f2);
87         delay_ms(1500);
88
```

```
88
89
90         delay_ms(100);
91         output_low(LED3);
92         delay_ms(100);
93         output_high(LED3);
94         delay_ms(100);
95         output_low(LED3);
96         delay_ms(100);
97         output_high(LED3);
98         delay_ms(100);
99         output_low(LED1);
100        output_low(LED2);
101        output_low(LED3);
102
103        // ----- Shutdown Ending -----
104
105        lcd_putc("\fALERT!");
106        printf("ALERT!\r\n");
107        delay_ms(1500);
108        // ----- Shutdown done -----
109        delay_ms(100);
110        output_low(LED1);
111        output_low(LED2);
112        output_low(LED3);
113        lcd_putc("\fShutdown Done!");
114        printf("Shutdown Done!\r\n");
115
116        output_low(f1);
117        output_low(f2);
118
119        while(TRUE);
120    }
121
122
```