

**L4 – EMBEDDED C
GENERATING A PROGRAM IN C**

PAGE: 1 OF 20

ASSIGNMENT TITLE : GENERATING A PROGRAM FOR A SPECIFIED
EMBEDDED REQUIREMENT IN C

MODULE : L4 – EMBEDDED C

MODULE CODE : ACDA4013

PROGRAMME : BENG ELECTRICAL & ELECTRONIC ENGINEERING

STUDENT NAME : FREDDIE WILSON

STUDENT NO : 2413880

**L4 – EMBEDDED C
GENERATING A PROGRAM IN C**

PAGE: 2 OF 20

TABLE OF CONTENTS

1	INTRODUCTION	4
2	BACKGROUND RESEARCH	4
3	CIRCUIT DESIGN	5
4	FLOWCHART	8
5	RESULTS	9
6	DETAILS OF TESTING	15
7	CONCLUSION	18
8	APPENDIX A – CIRCUIT TEST TABLE	19
9	REFERENCES	20

TABLE OF FIGURES

Figure 1: Proteus Circuit	5
Figure 2: Program Flowchart	8
Figure 3: Hex File	9
Figure 4: Defining Microcontroller Pins	9
Figure 5: Configuration Pin Directions	10
Figure 6: While SW1 Programme	11
Figure 7: While SW2 & SW3 Programmes	12
Figure 8: Fan Motor Running & Shutdown Sequence	13
Figure 9: Shutdown Messages	14
Figure 10: Switch 1 Pressed	15
Figure 11: Switch 2 Pressed	15
Figure 12: Switch 3 Pressed	16
Figure 13: System Alert	16
Figure 14: Fan Shutdown	17

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 3 OF 20

LIST OF ACRONYMS AND UNITS

Acronyms	Definition
A	Amperes
AC	Alternating Current
DC	Direct Current
Hz	Hertz
W	Watt
V	Volts
f	Frequency
F	Farad
Ω	Ohms
PWM	Pulse Width Modulation
DOL	Direct On Line
LED	Light Emitting Diode
LCD	Liquid Crystal Display
UART	Universal Asynchronous Receiver-Transmitter
PIC	Peripheral Interface Controller
I/O	Input / Output
ECU	Electrical Control Unit

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 4 OF 20

1 INTRODUCTION

This assignment involved developing a Proteus circuit and CCS C programme to simulate an automotive cooling system using the PIC16F877A microcontroller. Three switches represent real-world inputs — a temperature sensor, coolant flow transmitter, and engine start — with system responses displayed via LEDs, a 4-bit LCD, and UART communication.

2 BACKGROUND RESEARCH

According to Gimeno [1], automotive engines generate significant heat due to internal combustion and friction between moving components. To ensure maximum efficiency and maintain engine health, the engine temperature must be kept within the specified range. This is typically achieved by circulating engine coolant with a water pump, which absorbs the heat produced. The heated coolant then passes through the radiator, where a fan motor helps dissipate heat before the coolant is recycled through the engine [2].

These cooling systems are managed by the car's electronic control unit (ECU), which monitors engine temperature sensors and uses the feedback to adjust the cooling system when the specified threshold conditions are met [3]. These systems are also fitted with alert mechanisms, like dashboard lights or audible chimes, that warn the driver of abnormal or suboptimal conditions, such as overheating or insufficient coolant circulation [4].

The Proteus circuit mimics the conditional logic of automotive cooling systems, with each switch replicating a key engine event and each LED representing a dashboard indicator. A sequential control structure ensures each stage completes successfully before the next begins.

The PIC16F877A, with 33 digital I/O pins across five ports [5], is well-suited for this ECU-like application, enabling interfacing with LEDs, switches, a DC motor, an LCD, and UART. The motor uses microcontroller-driven mechanical control to activate the fan for a timed period when conditions are met.

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 5 OF 20

3 CIRCUIT DESIGN

The Proteus circuit, designed to meet the outlined requirements, is shown in Figure 1 below.

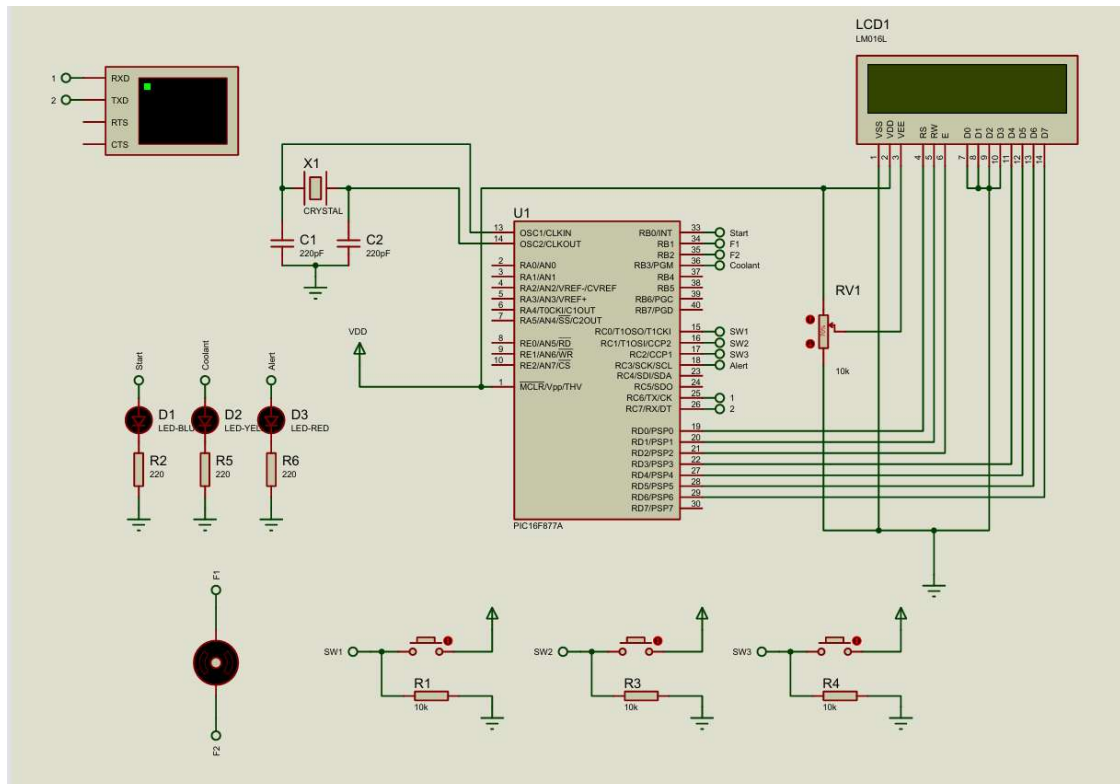


Figure 1: Proteus Circuit

Table I: Circuit Components

<u>Component</u>	<u>Tag</u>	<u>Function</u>	<u>Pin Connection</u>
PIC Microcontroller	U1	Processes switch inputs, executes the embedded C program, and controls LEDs, the DC motor, the LCD display, and system status messages through the UART.	-
Crystal Oscillator	X1	Provides a 20 MHz clock signal for the PIC microcontroller.	Connected to OSC1 (Pin 13) and OSC2 (Pin 14).

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 6 OF 20

220 pF Load Capacitor	C1	Used to stabilise the clock signal and works with C2.	Connected to OSC1 (Pin 13).
220 pF Load Capacitor	C2	Used to stabilise the clock signal and works with C1.	Connected to OSC2 (Pin 14).
LCD Display	LCD1	Displays operational system status messages, such as engine start, coolant ok, alerts, and shutdown.	Connected to PORTD, pins RD0 to RD2 for control and pins RD4 to RD7 for data.
10 kΩ LCD Potentiometer	RV1	Connected to the LCD's contrast pin, which adjusts the contrast of the characters by changing the voltage [6].	LCD (VEE).
Blue Start Indicator LED	D1	Illuminates when Switch 1 is pressed, and the engine start condition is active.	Connected to RB0.
Yellow Coolant Indicator LED	D2	Illuminates when Switch 2 is pressed, and the engine coolant is circulating.	Connected to RB3.
Red Alert Indicator LED	D3	Flashes three times to alert the user that the system is shutting down.	Connected to RC3.
220 Ω LED Current-Limiting Resistor	R2	Limits the current to the blue LED to prevent damage.	In series with RB0.
220 Ω LED Current-Limiting Resistor	R5	Limits the current to the yellow LED to prevent damage.	In series with RB3.
220 Ω LED Current-Limiting Resistor	R6	Limits the current to the red LED to prevent damage.	In series with RC3.
DC Motor	M1	Simulates the engine coolant fan, which is activated to reduce the engine coolant temperature.	Connected to RB1 and RB2.
Engine Start Switch	SW1	Simulates the engine start input, initiates the system, and allows further operation.	Connected to RC0.

**L4 – EMBEDDED C
GENERATING A PROGRAM IN C**

PAGE: 7 OF 20

Coolant Sensor Switch	SW2	Simulates a coolant temperature sensor, which must be met before the fan can be activated.	Connected to RC1.
Fan Activation Switch	SW3	Simulates the coolant temperature at which the fan must turn on.	Connected to RC2.
10 kΩ Pull-Down Resistor	R1	Ensuring output remains 'LOW' when the switch is open [7], guaranteeing a known logic level when not operated.	Connected to RC0.
10 kΩ Pull-Down Resistor	R3	Ensuring output remains 'LOW' when the switch is open [7], guaranteeing a known logic level when not operated.	Connected to RC1.
10 kΩ Pull-Down Resistor	R4	Ensuring output remains 'LOW' when the switch is open [7], guaranteeing a known logic level when not operated.	Connected to RC2.
UART Virtual Terminal	RXD / TXD	Displays serial messages received via UART from the PIC, allowing real-time system monitoring.	Connected to RC6 (TX), and RC7 (RX).
Power Supply	VDD / GND	Provides the circuit with the required operating voltage and reference ground for each component.	-

4 FLOWCHART

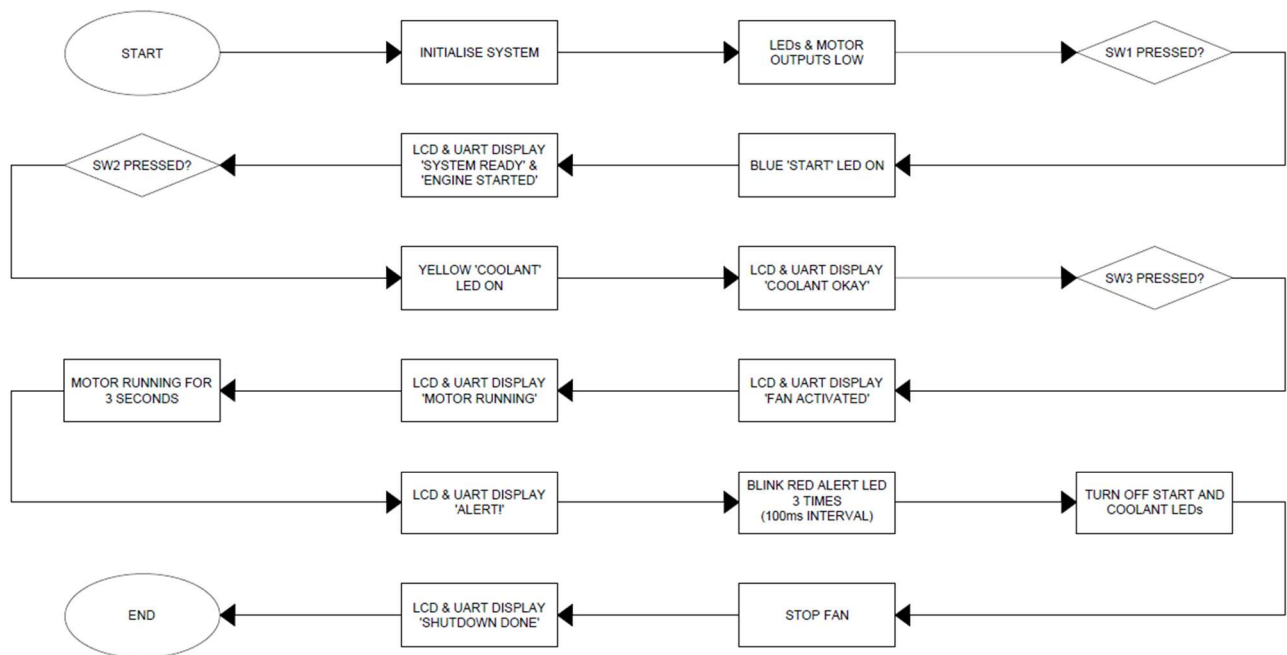


Figure 2: Program Flowchart

As shown in Figure 2, the program follows a sequential structure where each stage depends on the previous input completing successfully, with fan operation and LED indicators activating only when specified conditions are met. Flowchart symbols comply with ISO 5807:1985 [8].

L4 – EMBEDDED C GENERATING A PROGRAM IN C

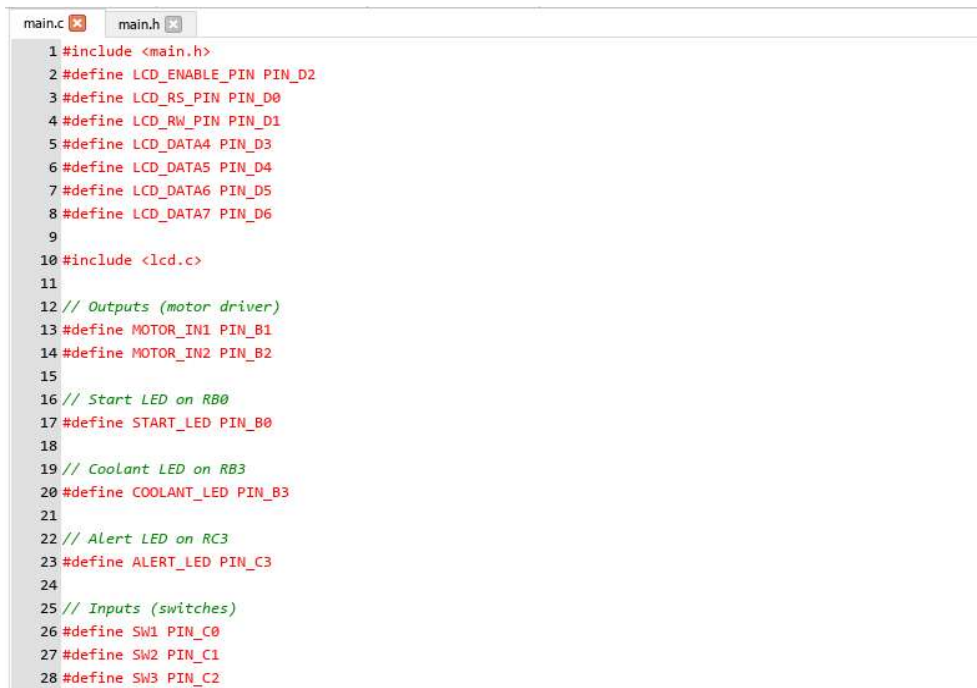
PAGE: 9 OF 20

5 RESULTS



```
main.c x main.h x
1 #include <16F877A.h>
2 #fuses NOWDT, HS, NOPROTECT, NOLVP
3 #use delay(crystal=20000000)
4 #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, bits=8, parity=N, stop=1)
5
6
```

Figure 3: Hex File



```
main.c x main.h x
1 #include <main.h>
2 #define LCD_ENABLE_PIN PIN_D2
3 #define LCD_RS_PIN PIN_D0
4 #define LCD_RW_PIN PIN_D1
5 #define LCD_DATA4 PIN_D3
6 #define LCD_DATA5 PIN_D4
7 #define LCD_DATA6 PIN_D5
8 #define LCD_DATA7 PIN_D6
9
10 #include <lcd.c>
11
12 // Outputs (motor driver)
13 #define MOTOR_IN1 PIN_B1
14 #define MOTOR_IN2 PIN_B2
15
16 // Start LED on RB0
17 #define START_LED PIN_B0
18
19 // Coolant LED on RB3
20 #define COOLANT_LED PIN_B3
21
22 // Alert LED on RC3
23 #define ALERT_LED PIN_C3
24
25 // Inputs (switches)
26 #define SW1 PIN_C0
27 #define SW2 PIN_C1
28 #define SW3 PIN_C2
```

Figure 4: Defining Microcontroller Pins

As shown in Figure 3, the program begins by including the main.h and lcd.c files and defining microcontroller pins. LCD control lines were assigned to D0–D2, data bits to D3–D6, and motor driver pins configured for clockwise rotation. The blue 'START' LED was mapped to B0, the yellow 'COOLANT OK' LED to B3, and the red 'ALERT' LED to C3.

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 10 OF 20

```
30 void main()
31 {
32     set_tris_d(0x00); // RD0-RD6 outputs to LCD
33     lcd_init();      // Initialise LCD once
34
35     // RB0-RB2 as outputs (blue LED + motor driver)
36     set_tris_b(0b1111000); // RB0, RB1, RB2 = outputs
37
38     // RC3 must be an output for the alert LED
39     set_tris_c(0b1111011); // RC3 = 0 (output), others inputs
40
41
42     // Ensure all outputs start LOW
43     output_low(PIN_B0); // Blue LED OFF
44     output_low(MOTOR_IN1);
45     output_low(MOTOR_IN2);
46     output_low(ALERT_LED); // Red LED OFF
47 }
```

Figure 5: Configuration Pin Directions

Furthermore, the input pins for each switch were assigned as follows: the first switch (SW1) to C0, the second (SW2) to C1, and the third (SW3) to C2.

The first line of the program, inside the void main loop, sets all PORTD pins as outputs by clearing every bit to 0 using the hexadecimal value 0x00. This means that these pins will drive the LCD display. The second line initialises the LCD, setting it to 4-bit mode and clearing the display.

The next instruction was to configure the PORTB outputs for the 'START' LED and the fan motor, leaving the other unused B pins as spare inputs for future use. Setting the port direction to 0b1111000 means pins RB7 to RB3 remain inputs ('1'), while pins RB2 to RB0 are outputs ('0').

The final direction of the pins to be set was for PORTC, which contains the UART, 3 switches, and the red 'ALERT' LED. This means all ports need to be left as inputs, except RC3. Therefore, set_tris_c (0b1111011) achieves this.

Setting all outputs to 'LOW' ensures a known safe state, preventing the 'START' or 'ALERT' LEDs from being falsely illuminated and preventing the fan motor from spinning, which could result in a dangerous state in a practical system.

```
48 // Wait for switches
49 while(!input(SW1));
50     delay_ms(50);
51
52 // Illuminate Blue Start LED
53 {
54     output_high(START_LED);
55
56 // System Ready
57 lcd_init();
58 lcd_putc("\fSystem Ready");
59 Printf("System Ready\r\n");
60 delay_ms(1000);
61
62 // Engine Started
63 lcd_putc("\fEngine Started");
64 Printf("Engine Started\r\n");
65 delay_ms(1000);
66
```

Figure 6: While SW1 Programme

The next section of code in the main loop is the while loop for SW1 while it is pressed. When the SW1 input is high, the blue 'START' LED output goes high, illuminating the LED indefinitely. In addition, the LCD display is initialised, reset, and then the message 'System Ready' is displayed, and the status message is sent to the virtual terminal in Proteus via UART. After that, there is a 1000 ms delay before the LCD and UART messages change, confirming that 'Engine Start' is present.

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 12 OF 20

```
67 while(!input(SW2));
68     delay_ms(50);
69
70     // Illuminate Yellow Coolant LED
71     {
72         output_high(COOLANT_LED);
73
74         // Coolant OK
75         lcd_putc("\fCoolant OK");
76         Printf("System Ready\r\n");
77         delay_ms(1000);
78
79         while(!input(SW3));
80             delay_ms(50);
81
82             // Fan Activated
83             lcd_putc("\fFan Activated");
84             Printf("Fan Activated\r\n");
85             delay_ms(1000);
86
87             // Motor Running
88             lcd_putc("\fMotor Running");
89             Printf("Motor Running\r\n");
90             delay_ms(1000);
```

Figure 7: While SW2 & SW3 Programmes

The previous code was repeated to illuminate the yellow coolant LED and display the 'Coolant OK' status update on the LCD and UART virtual terminal after the next switch, SW2, was activated. The LCD and UART also displayed 'Fan Activated' and 'Motor Running' after SW3 was pressed. A conservative 50 ms delay was applied after each button press to eliminate any effects of switch bounce in a practical system. However, this is not strictly necessary in a Proteus simulation.

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 13 OF 20

```
93 // Start fan
94 output_high(MOTOR_IN1);
95 output_low(MOTOR_IN2);
96
97 // Run fan for 3 seconds
98 delay_ms(3000);
99
100 // ALERT!
101 lcd_putc("\fALERT!");
102 Printf("ALERT!\r\n");
103 delay_ms(1000);
104
105 // Blink RED alert LED 3 times (100 ms interval)
106 for(int i = 0; i < 3; i++)
107 {
108     output_high(ALERT_LED);
109     delay_ms(100);
110     output_low(ALERT_LED);
111     delay_ms(100);
112 }
113 // Turn off Blue Start LED
114 {
115     output_low(START_LED);
116
117 // Turn off Yellow Coolant LED
118 {
119     output_low(COOLANT_LED);
120
121 // Stop fan
122 output_low(MOTOR_IN1);
123 output_low(MOTOR_IN2);
124 }
125 {
```

Figure 8: Fan Motor Running & Shutdown Sequence

The next lines of the program were used to start the fan motor. This was kept inside the while loop that ran while switch 3 was pressed. This was achieved by setting one of the motor input pins high and the other low, ensuring the motor rotated clockwise. The fan was then given a 3000 ms delay before the 'ALERT!' status update was displayed on the LCD and UART to alert the user. To accompany this message, the red LED was programmed to blink 3 times, with a 100 ms interval between blinks. Once the 'ALERT!' LED output was set to low, the start and coolant LEDs were also set to low, and both motor inputs were set to ensure a safe shutdown of the system.

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 14 OF 20

```
127 // Shutdown Done
128 lcd_putc("\fShutdown Done");
129 Printf("Shutdown Done\r\n");
130 delay_ms(1000);
131 }
132
133 // Idle forever
134 while(TRUE) { }
135
136
137 }
138 }
139 }
140 }
```

Figure 9: Shutdown Messages

The final lines of code were programmed to display a status message indicating a successful system shutdown on the LCD and UART displays. This was achieved using the `lcd_putc` and `Printf` commands. The system would then remain idle in this safe shutdown state indefinitely until each of the three switches was pressed to restart the event sequence.

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 15 OF 20

6 DETAILS OF TESTING

The Figures listed below contain screenshots of the Proteus circuit at each stage of its operation, from the start LED to full system shutdown.

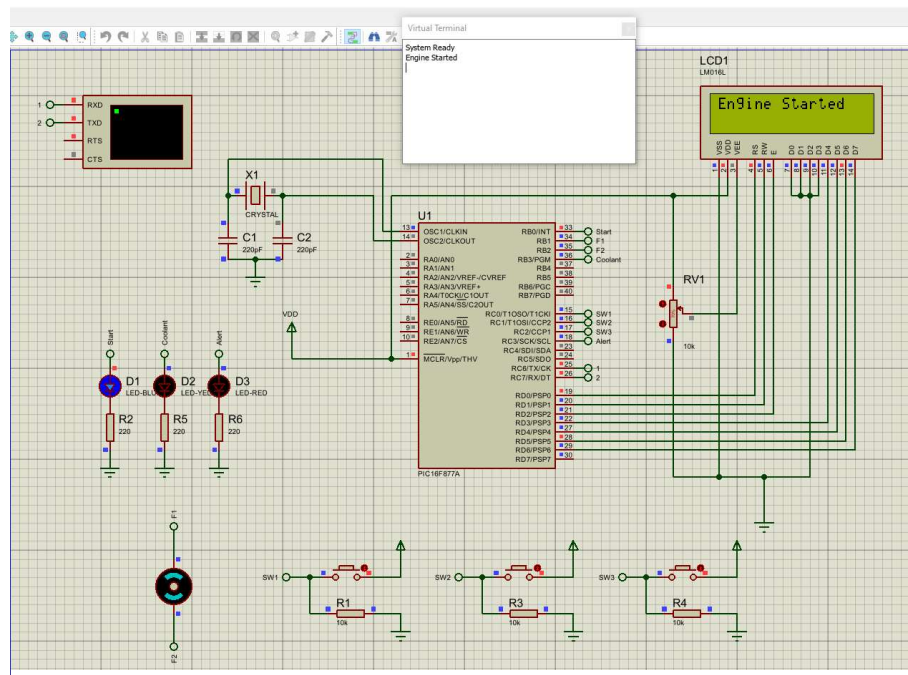


Figure 10: Switch 1 Pressed

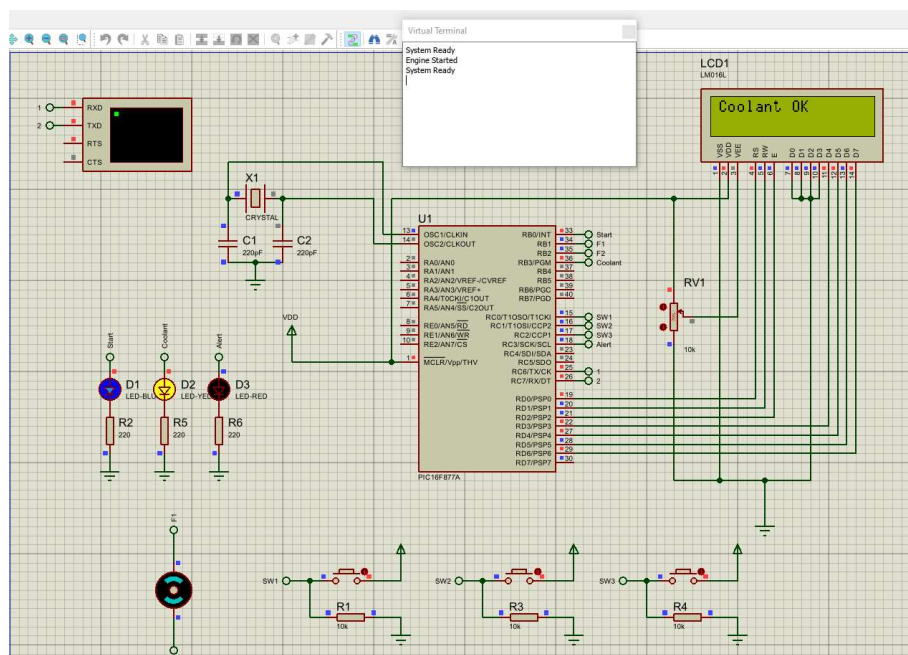


Figure 11: Switch 2 Pressed

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 16 OF 20

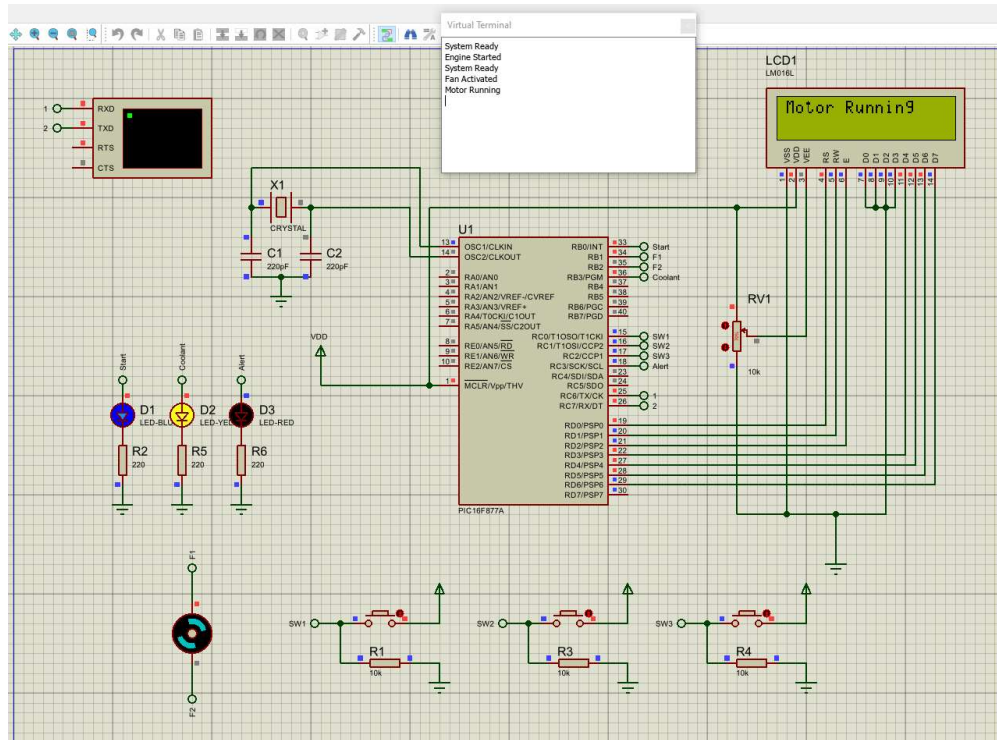


Figure 12: Switch 3 Pressed

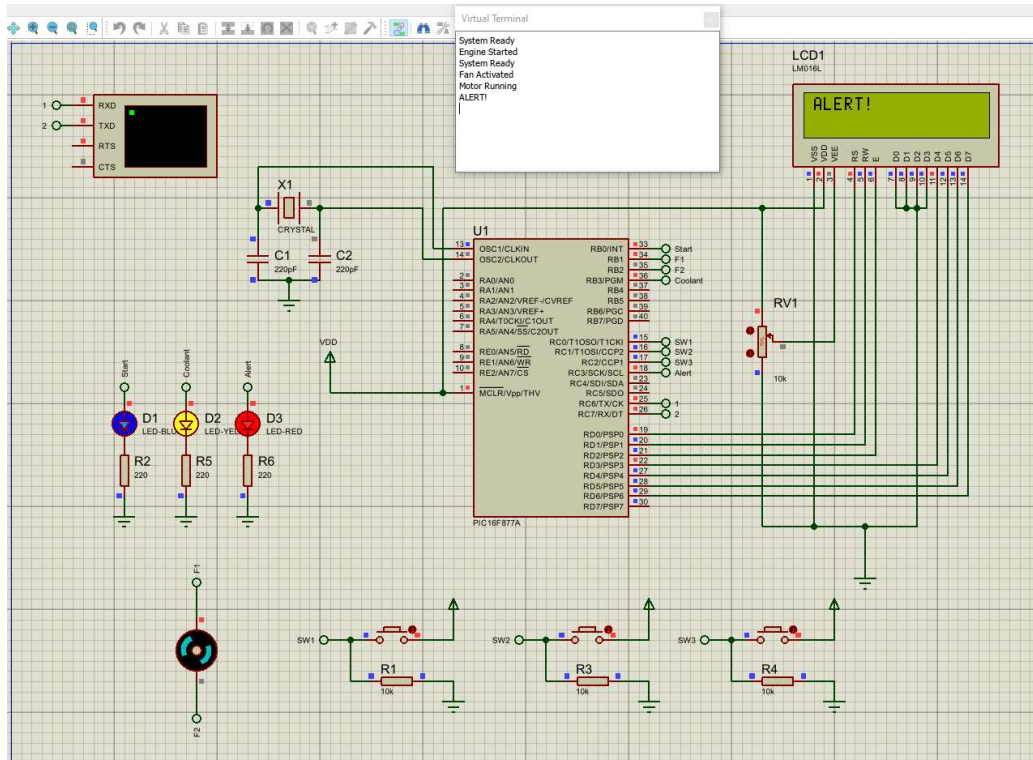


Figure 13: System Alert

L4 – EMBEDDED C GENERATING A PROGRAM IN C

PAGE: 17 OF 20

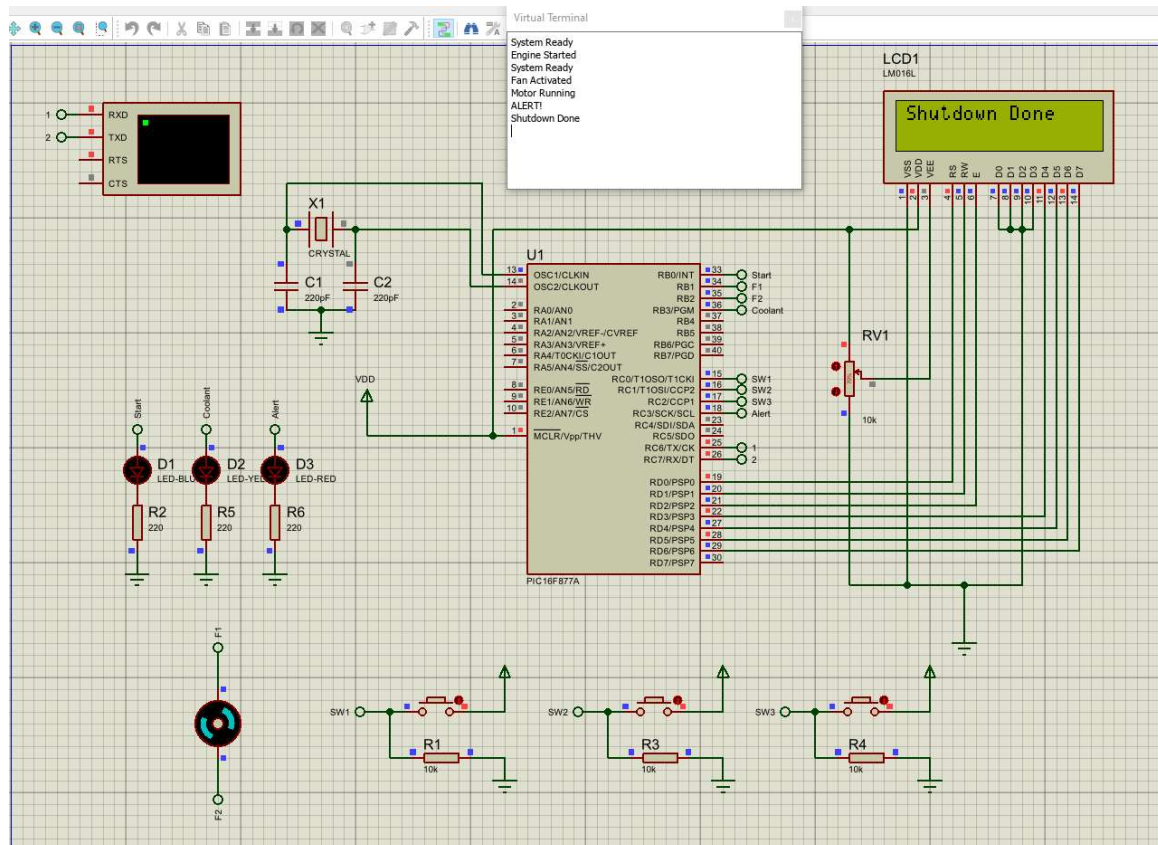


Figure 14: Fan Shutdown

Figures 10 through 14 show that the system operates as intended and meets all the outlined requirements.

During the initial testing, a few faults were identified. Firstly, the blue and yellow 'start' and 'coolant' LEDs remained illuminated even after the 'ALERT' LED flashed and the system shut down. It was discovered that this was because there wasn't originally an 'output low' line for each LED within the SW3 loop. After this was added, as seen in the screengrabs above, each LED turned off after the 'ALERT' LED flashed three times.

Moreover, another fault was identified in which the UART was not working. After reviewing the Printf lines, it became clear the issue was elsewhere, so the main.h file was checked. It was found that the UART compiler directive was missing. This was added to the hex file so that the microcontroller could send and receive serial data after setting the baud rate, transmitter and receiver pins, data bits, parity bit, and stop bit. Once this was set up, UART messages appeared on the virtual terminal as expected.

7 CONCLUSION

This project successfully replicated a vehicle cooling system using the PIC16F877A microcontroller and CCS C. The system followed a sequential operating sequence that included engine start-up, cooling and fan activation, and a safe shutdown. Visual LED feedback and system updates from the 4-bit LCD display and UART were used effectively to monitor each stage of operation.

Future system improvements could be made by replacing the pushbutton switches with temperature-sensor inputs to implement a consistent feedback loop between coolant temperature and the fan. Additionally, utilising PWM techniques to achieve variable fan speed, rather than a DOL start, would reduce mechanical strain on the fan and increase lifespan and efficiency.

**L4 – EMBEDDED C
GENERATING A PROGRAM IN C**

PAGE: 19 OF 20

8 APPENDIX A – CIRCUIT TEST TABLE

Table A.I: System Test Results

<u>Test Stage</u>	<u>Input Condition</u>	<u>Expected Output</u>	<u>Actual Output</u>
System Off	No Switches Pressed	All LEDs OFF, Motor OFF, LCD & UART Clear/Idle	As expected
Engine Start	SW1 Pressed	Blue LED ON, LCD & UART: "System Ready", "Engine Start"	As expected
Coolant Confirmation	SW2 Pressed	Yellow LED ON, LCD & UART: "Coolant OK"	As expected, except UART wrongly showed "System Ready".
Fan Activation	SW3 Pressed	Motor runs for 3s, LCD & UART updates	As expected
Alert	After 3 seconds	Red LED blinks 3 times (100 ms intervals)	As expected
Shutdown	Post Alert	All LEDs & Motor OFF, LCD & UART: "Shutdown Done"	As expected

9 REFERENCES

- [1] P. Gimeno, "Engine cooling system: how it works and main components," Industrias Dolz, May 27, 2024. <https://www.idolz.com/en/engine-cooling-system-how-it-works-and-main-components/> (accessed Apr. 13, 2026).
- [2] K. Nice, "How Car Cooling Systems Work," HowStuffWorks, Feb. 11, 2021. <https://auto.howstuffworks.com/cooling-system.htm> (accessed Apr. 13, 2026).
- [3] "A complete guide to engine control units - Halfords," Halfords.com, Oct. 02, 2020. <https://blog.halfords.com/a-complete-guide-to-engine-control-units/> (accessed Apr. 13, 2026).
- [4] J. Mournian, "Coolant Warning Light: What You Should Do," Stoneacre Motor Group, May 16, 2024. <https://www.stoneacre.co.uk/blog/coolant-warning-light-what-you-should-do> (accessed Apr. 13, 2026).
- [5] "PIC16F87XA Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers." Available: <https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf>
- [6] "LCD 16x2 Pinout, Commands, and Displaying Custom Character," Electronics For You – Official Site ElectronicsForU.com, Aug. 25, 2023. <https://www.electronicsforu.com/technology-trends/learn-electronics/16x2-lcd-pinout-diagram#:~:text=An%20LCD%20screen%20is%20an,15> (accessed Apr. 13, 2026).
- [7] R. Keim, "Pull-up and Pull-down Resistors," Eepower.com, 2026. <https://eepower.com/resistor-guide/resistor-applications/pull-up-resistor-pull-down-resistor/#> (accessed Apr. 13, 2026).
- [8] O. for, "ISO 5807:1985," ISO, 2026. <https://www.iso.org/standard/11955.html> (accessed Apr. 13, 2026).