

Technical Report – Embedded C – ACDA4013

Electrical & Electronic Engineering Degree Apprenticeship

Joshua H – Student Number: 2420839

Contents

Introduction	3
Background Research	3
Circuit Design	4
Function Diagram	5
Simulation.....	7
Testing	11
Conclusion.....	14
References	15
Appendix.....	16

Introduction

Embedded systems are used throughout modern life. They are a specialised electronic system dedicated to a specific function within a larger system[1]. Within the automotive sector, embedded systems are fundamental to vehicle function and control. The advent of engine control units (ECU) “revolutionized the automotive industry”[1], replacing earlier mechanical systems. Initially produced to control engine, transmission and alert functions, these embedded systems are now far more sophisticated tracking sensor outputs, fuel consumption, and numerous other parameters to improve performance, user experience and safety [1, 2]. Modern vehicles rely on electronic monitoring of temperature levels to prevent engine damage; microcontrollers are used to manage sensor feedback and active cooling components such as the radiator fan. Simulating these operations offers insight into real-world automotive mechanisms. This technical report documents the design and simulation of an automotive cooling system using the PIC16F877A microcontroller. The project replicates the sequence of a simple coolant system. Incorporating switches for sensor feedback, a DC motor to represent radiator fans, LED and LCD indicators for user feedback. The report details the design implementation and simulation results of the cooling system model.

Background Research

Embedded systems are critical to running a safe, reliable and efficient vehicle. Embedded systems in vehicles are specialised circuits designed to carry out a set number of functions ranging from safety-critical to user comfort. Common examples of embedded systems within a vehicle include Infotainment systems, GPS navigation, diagnostic systems, safety systems, and the ECU [3]. The ECU is the main controller of an array of embedded systems within a vehicle. Using variable voltages and frequency data it can compare to optimum values stored in memory to affect change within the operation of the vehicle. [4]

In an internal combustion engine, a coolant system is not only necessary for the engine to function effectively but also to increase the service life, fuel economy, and reduce emissions outputted by the car. (Science Direct). Generally, coolant systems work in two ways: mechanically controlled and electrically controlled also known as map-controlled. Mechanically controlled coolant systems are through a temperature-sensitive wax valve which opens to different degrees based on the coolant temperature. [5]. Map-controlled systems however work through a temperature sensor most commonly being either a thermistor, resistive temperature detector, thermocouple or integrated circuits. [1,2,6]. The working principle of these components being that the sensor converts the coolant temperature into a voltage which the ECU then uses to manage the coolant temperature efficiently. “The voltage is higher when the engine is cool and the voltage is lower when the engine is hot.”[1]. Based on this variable voltage the ECU can control when fans should and should not spin.

Circuit Design

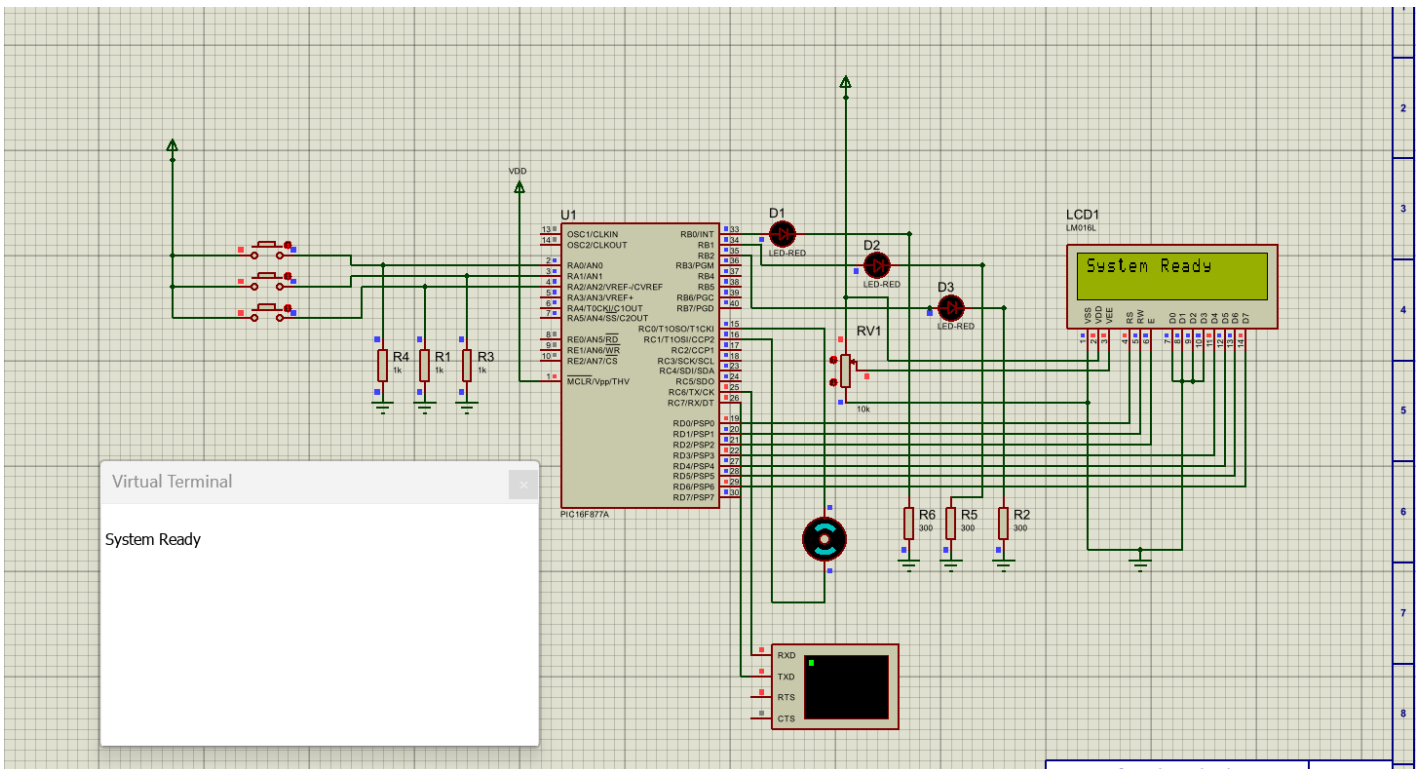
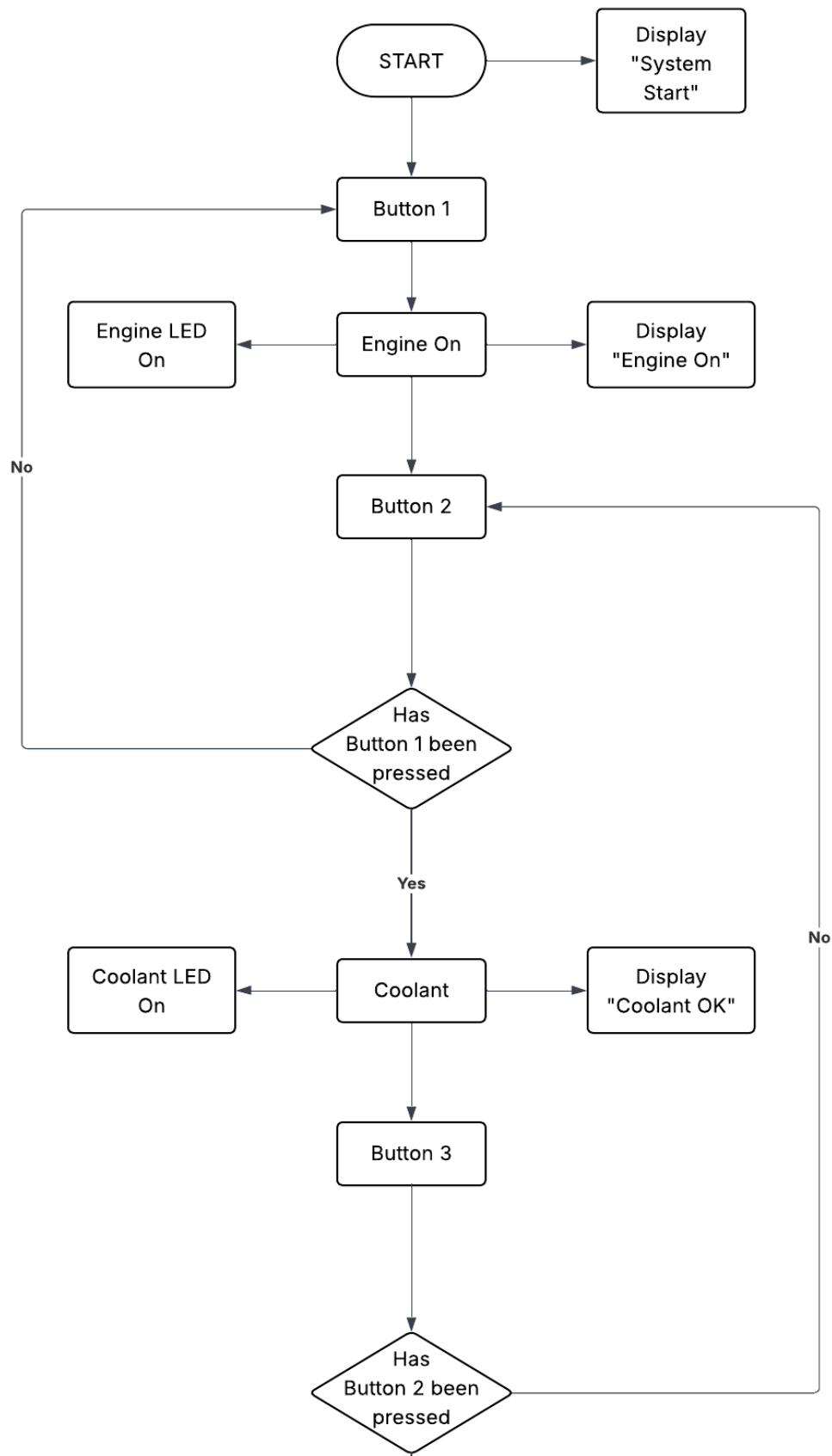


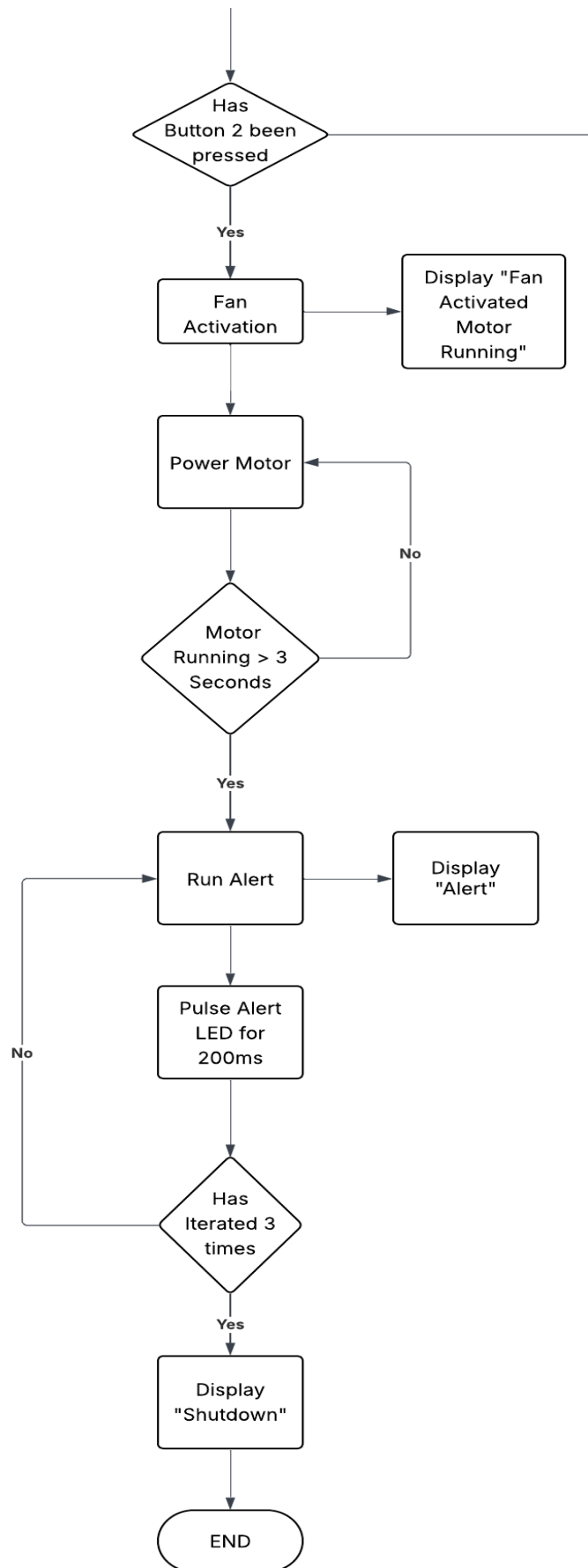
Figure 1

The circuit designed, in Proteus 9, replicates the embedded system that controls the radiator fans in an ICE coolant system. The three buttons are inputs with C representing a temperature sensor reading that the microcontroller would read as too high, this should cause the radiator fans to rotate. However, before the microprocessor is able to control the engine should be on and the coolant should read ok. These two statements must be satisfied.

The PIC16F877A microcontroller has been chosen for its ability to perform the functions required by this project's brief. With four 8-bit bidirectional ports, it can manage an LCD display, drive a simple DC motor, and operate three LEDs simultaneously. In addition, it supports USART communication, allowing it to transmit and receive serial data, making it suitable for integration with a virtual terminal during simulation. The circuit is built around this microcontroller like most embedded systems. The switches connected to Port A act as simple sensors, for the microcontroller to respond to and the LEDs and LCD work as outputs for the users, mirroring a car. The DC motor represents the radiator fans used to provide greater airflow when the car is stationary or overheating[source].

Function Diagram





The flowchart above outlines the systems function from start to finish. On system-on the microcontroller sends signals via UART to the serial monitor and via SPI to the LCD. These signals should display “System Start”. Then once the first button is pressed the microcontroller should send signals to display “Engine on” and raise an output pin high to turn a LED on. This LED should stay on for the entire operation. Once the second button is pressed the microcontroller should check to see if the first button flag has been raised. If so, the microcontroller may move on to displaying “Coolant Ok” and lighting another LED. If not, however the microcontroller should do nothing. When the final button has been pressed and the appropriate flag has been raised the motor will be powered as well as “Fan Activated Motor Running” displayed. This will continue for three seconds, the microcontroller timing the event to ensure accuracy. After three seconds the motor will power down, and a LED will flash for 200ms cycles 3 times, “Alert” will be displayed and following this the system will display “Shutdown”.

Simulation

Below are screenshots of the simulation running in Proteus 9.

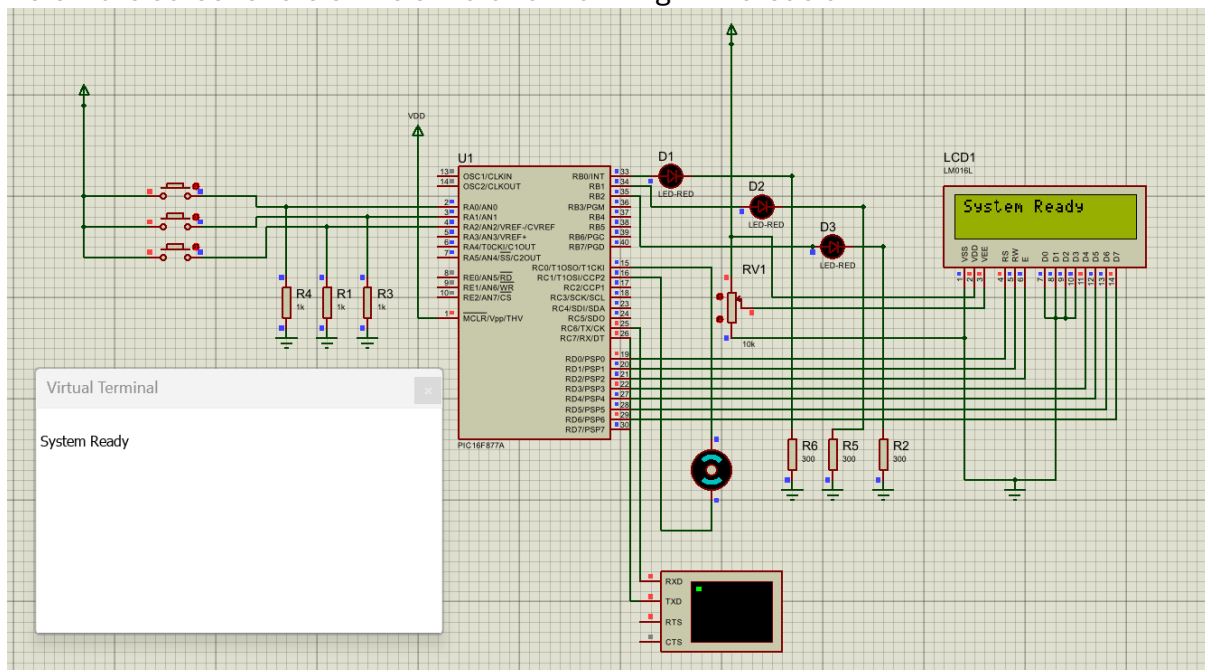


Figure 2

Figure 2 shows the initial state of the program. All ports have been set low, ports C and D control the rs232 and LCD communication respectively so after initialisation the pins on these ports have been driven high or low to transmit any data necessary.

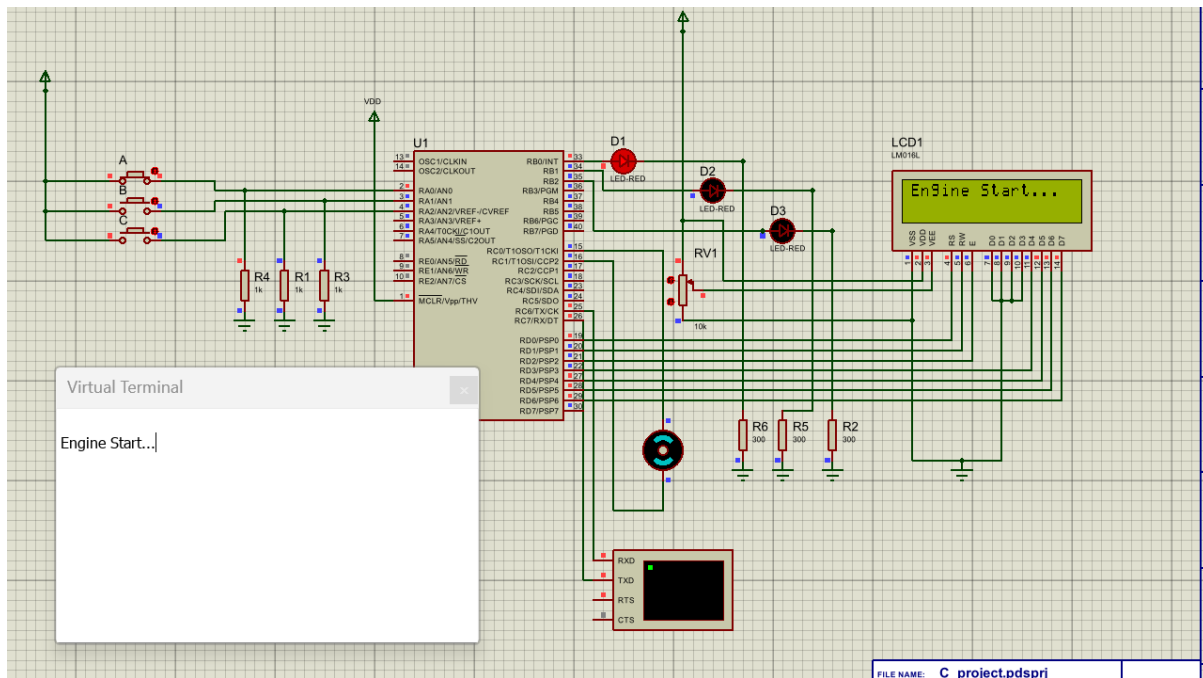


Figure 3 shows the first stage in the program, after switch A has been pressed, this causes pin A9 to be driven high and as the button has not been pressed on previously the if statement on line 49 is satisfied. This results in the LED (D1) to light as well as causing the LCD and virtual terminal to display “Engine start...”.

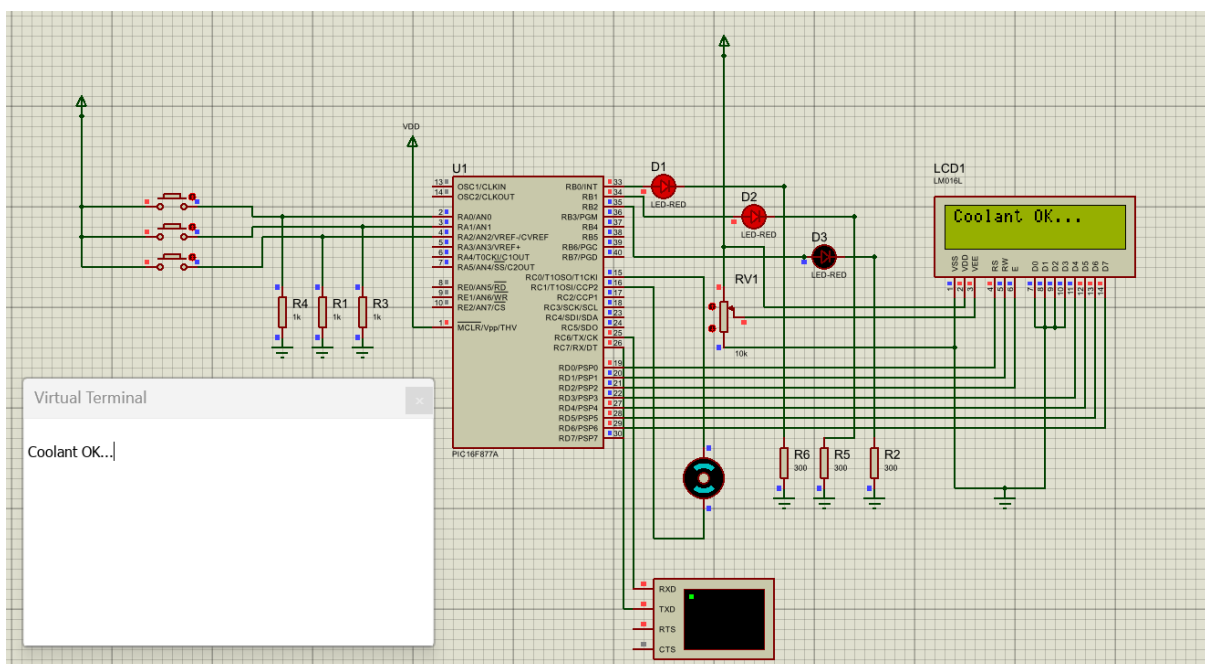


Figure 3 shows the second stage in the program, once the button A has been pressed inputs to button B can be processed, once this happens LED D2 lights, the LCD and virtual terminal then output “Coolant OK”.

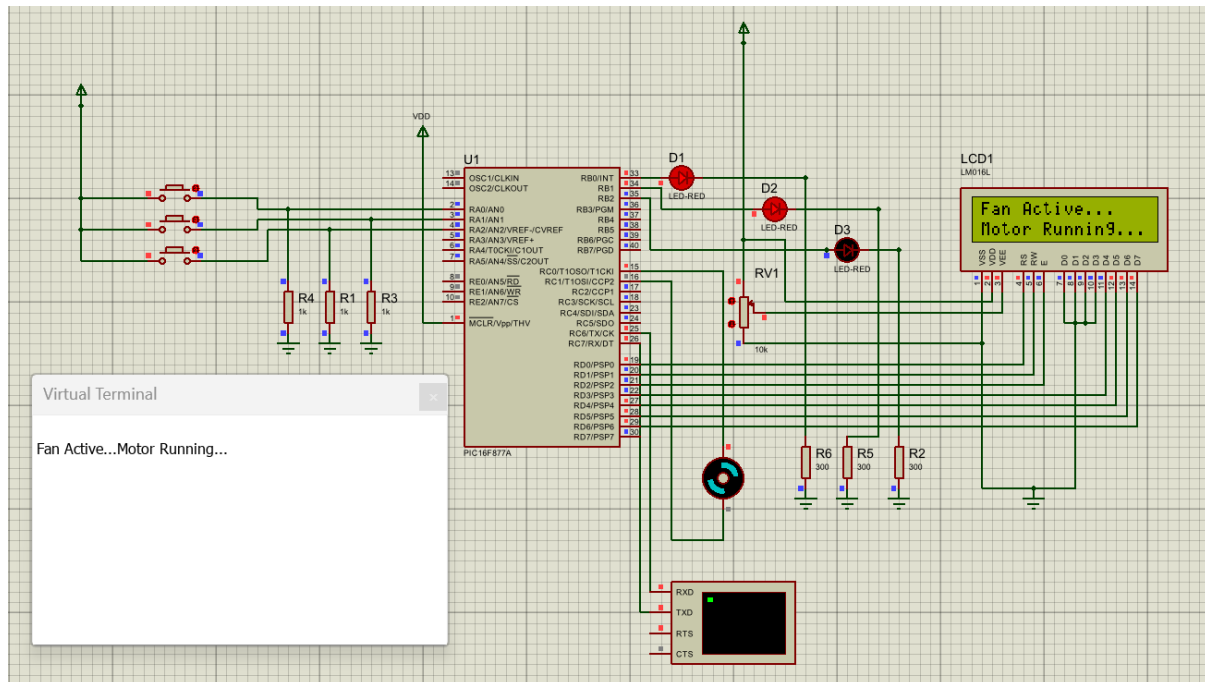


Figure 5

Figure 4 shows the beginning of the third stage, once button 3 and the two buttons before it have been pressed pin C6 is driven high and pin C7 is driven low to allow the motor to spin the program is then held in a delay for three seconds to allow the motor to complete its cycle. The LCD and virtual terminal are also updated.

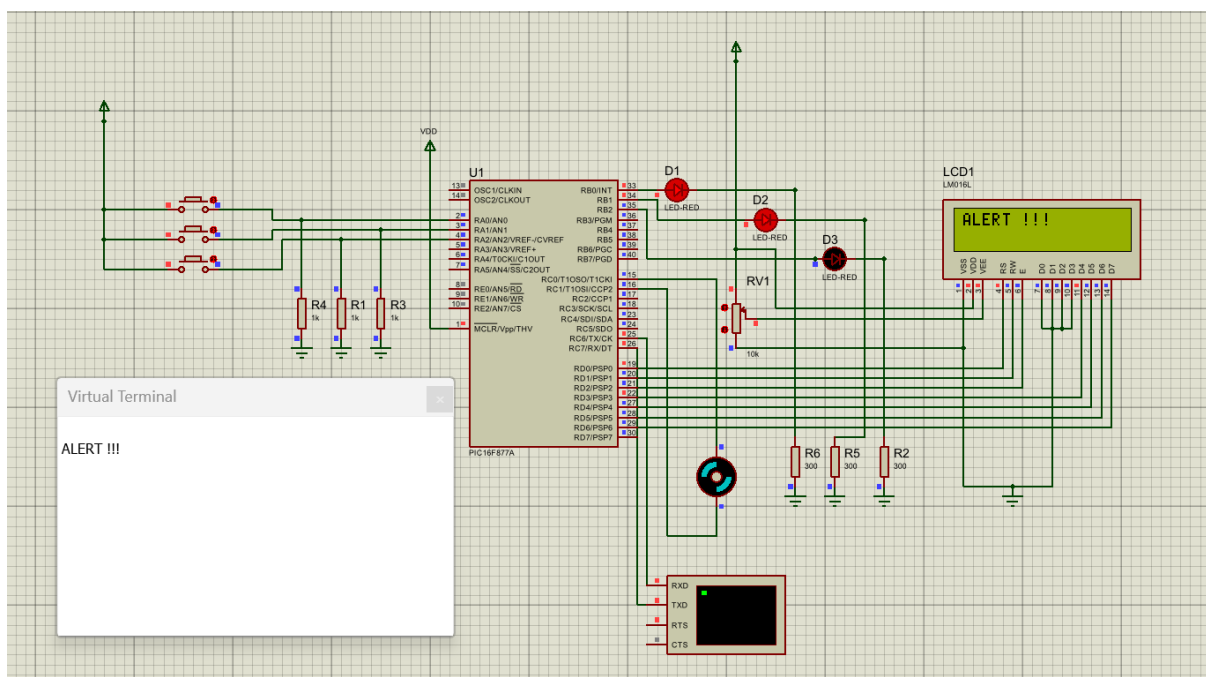


Figure 6

Figure 5 shows the end of the third stage. Pin C six is brought low and the motor stops rotating. D3 is then toggled on and off through a while loop 3 times, during which the LCD and virtual terminal are updated to read "ALERT !!!".

Testing

Testing will be made up of four parts, to test the functionality of the circuit. Testing outlined below will be completed chronologically through the program, mirroring the flowchart above.

Test Number	Test Spec	Expected	Actual	Adjustments
1.1	On start LCD should display "system ready"	When the program is started the LCD displays "system ready"	The LCD displays "system ready"	None required
1.2	On start the virtual terminal should display "system ready"	When the program is started the virtual terminal displays "system ready"	the virtual terminal displays "system ready"	None required
1.3	Pressing button 2 should have no effect	Nothing changes	Nothing changes	None required
1.4	Pressing button 3 should have no effect	Nothing changes	Nothing changes	None required
2.1	When button 1 is pressed LED 1 should light and stay lit	When button 1 is pressed LED 1 should light	LED 1 lights and stays on	None required
2.2	When button 1 is pressed the LCD should output "Engine On" and the old output should clear	When button 1 is pressed "Engine On" should appear and be the only text on the LCD	The LCD displays "Engine On"	None required
2.3	When button 1 is pressed the virtual terminal should output "Engine On" and the old output should clear	When button 1 is pressed "Engine On" should appear and be the only text in the virtual terminal	The virtual terminal displays "Engine On"	None required

3.1	When button 2 is pressed LED 2 should light and stay lit	When button 2 is pressed LED 2 should light	LED 2 lights and stays on	None required
3.2	When button 2 is pressed the LCD should output “Coolant Ok” and the old output should clear	When button 2 is pressed “Coolant Ok” should appear and be the only text on the LCD	The LCD displays “Coolant Ok”	None required
3.3	When button 2 is pressed the virtual terminal should output “Coolant OK” and the old output should clear	When button 2 is pressed “Coolant Ok” should appear and be the only text in the virtual terminal	The virtual terminal displays “Coolant Ok”	None required
3.4	Pressing button 1 should have no effect	Nothing changes	Nothing changes	None required
4.1	When button 3 is pressed the motor should run for three seconds	The motor runs for three seconds after the button has been pressed	The motor runs for three seconds	None required
4.2	When button 3 is pressed the LCD should output “Fan Active Motor Running” and the old output should clear	When button 3 is pressed “ Fan Active Motor Running” should appear and be the only text on the LCD	The LCD displays “Fan Active Motor Running”	None required
4.3	When button 3 is pressed the virtual terminal should output “ Fan Active Motor Running ” and the old output should clear	When button 3 is pressed “ Fan Active Motor Running ” should appear and be the only text in the virtual terminal	The virtual terminal displays “Coolant Ok”	None required
4.4	LED 3 pulses 3 times after the motor turns of	Once the motor has finished its cycle LED 3	LED 3 blinked 3 times after the	None required

		should blink 3 times	motor had switched off	
4.5	The LCD should output "ALERT" and the old output should clear after the motor is off	Once the motor has finished its cycle the LCD should output "ALERT"	The LCD displays "ALERT"	None required
4.6	The virtual terminal should output "ALERT" and the old output should clear after the motor is off	Once the motor has finished its cycle the virtual terminal should output "ALERT"	The virtual terminal displays "ALERT"	None required
4.7	Once LED 3 has finished its cycle the LCD should display "Shutdown"	The LCD should output "Shutdown" after the led has finished cycling	The LCD displays "Shutdown"	None required
4.8	Once LED 3 has finished its cycle the virtual terminal should display "Shutdown"	The virtual terminal should output "Shutdown" after the led has finished cycling	The virtual terminal displays "Shutdown"	None required

Conclusion

In conclusion, the report successfully demonstrates an abstraction of an automotive cooling system using the PIC16F877A microcontroller and the development language C. With the use of switch-based inputs, LED indicators, LCD feedback, UART communication and a DC motor, the system built in Proteus can effectively replicate the stages of a simplified coolant system. The coded logic ensures that operations can only happen in the order specified by the specification, reflecting the behaviour of real-world embedded systems. Results from Proteus confirm outputs behave as expected based on given inputs, with testing verifying expected behaviour matched actual results with no deviations. Overall, the report illustrates how microcontrollers can be used to manage sensor input, execute sequential conditional operations and provide user feedback.

Future improvements to the design could include using a potentiometer to represent a thermistor commonly used in coolant systems, into one of the microcontrollers ADC inputs. The variable voltage input could then be compared to a hardcoded look-up table which would allow the system to respond dynamically to changing sensor inputs. Having dynamic responses to sensor inputs will allow the system, possibly through pulse width modulation, to increase or decrease the speed of the motor depending on how “hot” or “cold” the coolant is. Another improvement could be to increase the complexity of the coolant system. The heater matrix for example is missing from the system. This component uses coolant heat to create warm air for the car’s cabin [8], in turn this lowers the coolant temperature. Implementing this change dramatically increases the complexity of the system and would require interrupt driven operations which are lacking in the code base. Expanding the system to include these features allows the system to reflect the complex automotive cooling system more closely. [7]

References

[1]

I. Ahmed, "Engine Coolant Temperature Sensor in Automotive Applications," *ResearchGate*, Jun. 30, 2020. https://www.researchgate.net/publication/344327217_Engine_Coolant_Temperature_Sensor_in_Automotive_Applications

[2]

K. Thermometrics, N. El-Awar, D. Geer, T. Krellner, and P. Straub, "AUTOMOTIVE TEMPERATURE SENSING," 1999. Accessed: Dec. 15, 2025. [Online]. Available: <http://denethor.wlu.ca/pc300/projects/library/autoapp.pdf>

[3]

A. Mountford, "Embedded Systems in Vehicles: A Comprehensive Guide | T-Plan," *T-Plan*, Feb. 04, 2025. https://www.t-plan.com/blog/exploring-embedded-systems-in-vehicles-a-comprehensive-guide/#elementor-toc__heading-anchor-1 (accessed Dec. 15, 2025).

[4]

Denise, "How a Car Thermostat Works | Blog," *German Automotive*, May 24, 2021. <https://germanauto.co.uk/how-a-car-thermostat-works/>

[5]

D. Palle, R. Ramadevi, and R. Kanchi, "Design and Development of Microcontroller Based Engine Cooling System for Auto Mobile Applications," *International Journal of Engineering Innovation & Research*, vol. 6, no. 5, pp. 2277–5668, 2017, Accessed: Dec. 15, 2025. [Online]. Available: https://ijeir.org/Download/conference/ICAEMS-17/42_ICTR6001_JOURNAL_FINAL.pdf

[6]

H. Basir, S. A. Hosseini, S. Nasrollahnezhad, A. Jahangiri, and M. A. Rosen, "Investigation of engine's thermal management based on the characteristics of a map-controlled thermostat," *International Communications in Heat and Mass Transfer*, vol. 135, no. 135, p. 106156, Jun. 2022, doi: <https://doi.org/10.1016/j.icheatmasstransfer.2022.106156>.

[7]

M. Mohamed, M. Shedid, El-Demerdash, and M. Fatouh, "Performance of Electronically Controlled Automotive Engine Cooling System Using PID and LQR Control Techniques," *IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)*, vol. 15, no. 3, pp. 42–51, 2018, doi: <https://doi.org/10.9790/1684-1503034251>.

[8]

"Heater core," *Wikipedia*, Aug. 13, 2023. https://en.wikipedia.org/wiki/Heater_core

Appendix

```
main.c*
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102

if(input(PIN_A2) == 1 & (button1 == 1) & (button2 == 1)){ // If Pin A2 goes high and all flags have been raised

    lcd_putc("\fFan Active...\nMotor Running..."); // Print to LCD
    printf("\fFan Active...\nMotor Running..."); // Print to Terminal

    output_high(MOTOR_P); // Run the motor
    output_low(MOTOR_N);
    delay_ms(3000); // For three seconds
    output_low(MOTOR_P); // Stop the motor

    int i = 0; // Set iterable

    while( i < 3){ // Repeat 3 times

        lcd_putc("\fALERT !!!"); // Print to LCD
        printf("\fALERT !!!"); // Print to Terminal

        output_high(LED2); // Flash LED 2
        delay_ms(200);
        output_low(LED2);
        delay_ms(200);

        i += 1; // Iterate
    }

    lcd_putc("\fShutdown"); // Print to LCD
    printf("\fShutdown"); // Print to Terminal
}

main.c*
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

void main(){ // Setup

    set_tris_a(0x10); // Set Port A to input

    set_tris_b(0x00); // Set Port B to output
    set_tris_c(0x00); // Set Port C to output
    set_tris_d(0x00); // Set Port D to output

    // Initailising flags
    int button1 = 0;
    int button2 = 0;

    // Setting all I/O pins to 0
    output_a(0);
    output_b(0);
    output_c(0);
    output_d(0);

    // Initailising the LCD
    lcd_init();

    // Printing to Terminal and LCD
    lcd_putc("\fSystem Ready");
    printf("\fSystem Ready");
}
```

```

main.c*
52 printf("\fSystem Ready");
53
54
55 while(TRUE) // Loop forever
56 {
57
58
59 if(input(PIN_A0) == 1 & button1 == 0){ // If Pin A0 goes high and button 1 hasnt be pressed
60
61     lcd_putc("\fEngine Start..."); // Print to LCD
62     printf("\fEngine Start..."); // Print to Terminal
63     output_high(LED0); // Light LED 1
64     button1 = 1; // Update flag, button 1 has been pressed
65 }
66 if((input(PIN_A1) == 1) & (button1 == 1) & (button2 == 0)){ // If Pin A1 goes high, button 1 has already been pressed,
67 // and button 2 hasnt been pressed
68     lcd_putc("\fCoolant OK..."); // Print to LCD
69     printf("\fCoolant OK..."); // Print to Terminal
70     output_high(LED1); // Light LED 2
71     button2 = 1; // Update flag, button 2 has been pressed
72 }
73
74 if(input(PIN_A2) == 1 & (button1 == 1) & (button2 == 1)){ // If Pin A2 goes high and all flags have been raised
75
76     lcd_putc("\fFan Active...\nMotor Running..."); // Print to LCD
77     printf("\fFan Active...\nMotor Running..."); // Print to Terminal
78 }
79 }
80 }

```

```

main.c*
1 #include <16F877A.h>
2 #fuses NOWDT, HS, NOPROTECT, NOLVP
3
4 // Setting program clock speed
5 #use delay(clock=2000000)
6
7 // Setting serial communications
8 #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, bits=8, parity=N, stop=1)
9
10 // Pin Name Definitions
11 #define LED0 PIN_B0
12 #define LED1 PIN_B1
13 #define LED2 PIN_B2
14
15 #define MOTOR_P PIN_C0
16 #define MOTOR_N PIN_C1
17
18 // These definitions are required for the LCD library to function
19 #define LCD_ENABLE_PIN PIN_D2
20 #define LCD_RS_PIN PIN_D0
21 #define LCD_RW_PIN PIN_D1
22 #define LCD_DATA4 PIN_D3
23 #define LCD_DATA5 PIN_D4
24 #define LCD_DATA6 PIN_D5
25 #define LCD_DATA7 PIN_D6
26 #include <lcd.c>
27

```