



Y Drindod Dewi Sant
University of Wales
Trinity Saint David

EMBEDDED C

ACDA4013

Conner Kinsey (2415170)

Contents

1. Introduction	2
2. Background research.....	3
3. System design	4
3.1 Microcontroller	5
3.2 Oscillator clock	6
3.3 Input switches	7
3.4 LEDs	8
3.5 LCD	9
3.6 UART	10
3.7 Motor simulation.....	11
4. Program logic	12
4.1 Software implementation	13
5. Simulation results	16
6. Discussion.....	16
7. Conclusion	17
8. References	18

1. Introduction

This project presents the design and simulation of an automotive cooling system status simulator using the PIC16F877A microcontroller. The system models an engine cooling process, including engine start, coolant monitoring, fan activation and system shutdown. The design is implemented in Proteus and programmed in CCS C, allowing the behaviour of both hardware and software to be put into use.

User inputs are simulated using push-button switches, while system outputs are displayed through LEDs, an LCD module, and UART communication. This project demonstrates how embedded systems can be used to control real world processes in automotive applications.

2. Background research

Modern vehicles rely heavily on embedded electronic systems to monitor engine conditions. They also control supporting subsystems such as cooling, fuel delivery and safety mechanisms [1]. One of these subsystems is the engine cooling system, which ensures that the engine operates within a safe temperature range. If this fails, the engine will not maintain an appropriate temperature that can result in overheating and potential engine damage [2].

In this project, the behaviour of these sensors is simplified using push-button inputs. This approach allows the operation of the system to be demonstrated clearly and effectively.

The PIC16F877A microcontroller is used, it is an 8-bit microcontroller that provides multiple digital input/output ports, timers and a USART. This allows data to be transmitted to a virtual terminal.

3. System design

The circuit was designed and implemented in Proteus using the PIC16F877A microcontroller as the central control unit. The schematic diagram (see Figure 1) shows the complete system including input switches, output indicators and the LCD display.

PORTD is used for interfacing with the LCD in 4-bit mode. The LCD provides user-friendly visual feedback also connected to UART, displaying messages such as system status and operational stages.

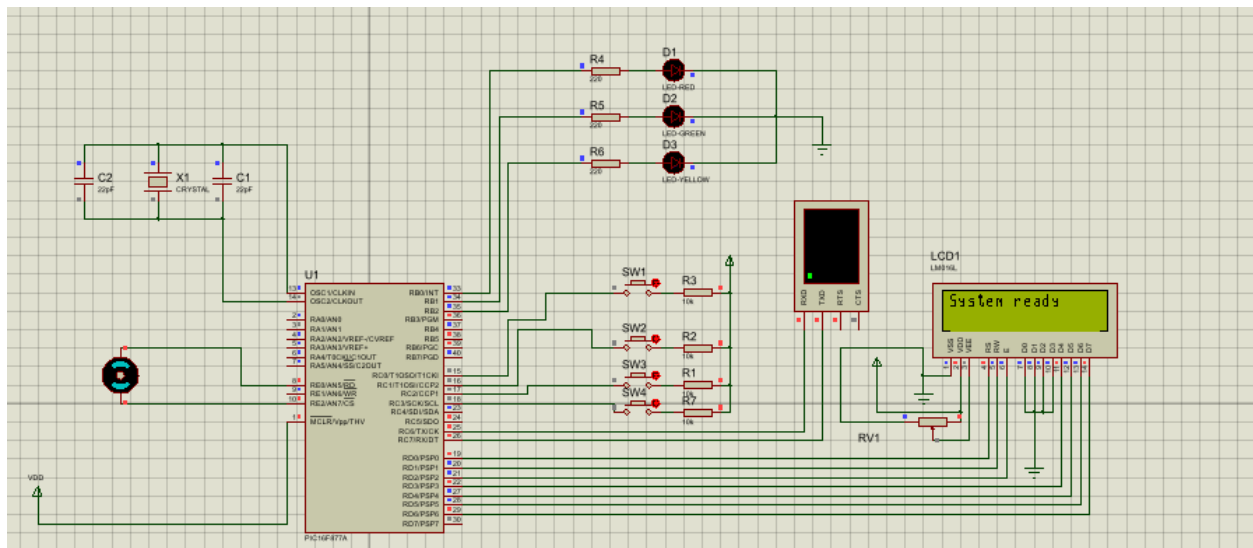


Figure 1: Schematic diagram of the circuit.

3.1 Microcontroller

The PIC16F877A was selected to be used as the microcontroller for this project. It has five input and output ports PORTA, PORTB, PORTC, PORTD, and PORTE. Each port can be configured to requirements and either used as analog or digital inputs [3].

As the microcontroller can process both inputs and outputs, it can control multiple outputs including LEDs for status indication, a 16×2 LCD for real time message display and a DC motor to represent the radiator cooling fan.

Also the microcontroller has internal timers, which is used manage delays, such as the fan operation period and LED alert blinking sequence [3].

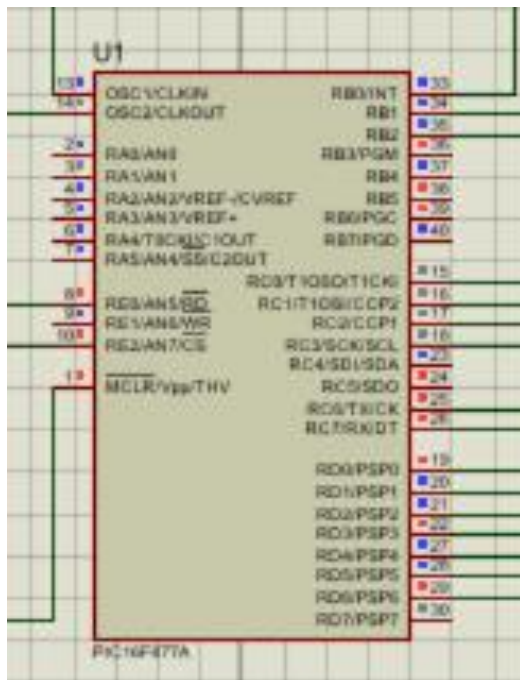


Figure 2: PIC16F877A microcontroller.

3.2 Oscillator clock

As the PIC16F877A hasn't got an internal oscillator [3], it has been configured with an external crystal oscillator. The oscillator has been set at 20MHz clock time, connected across OSC1 and OSC2 pins. This is connected with two 22 pF capacitors C1 and C2. This provides a stable clock signal, which is essential for reliable microcontroller operation and timing accuracy [4].

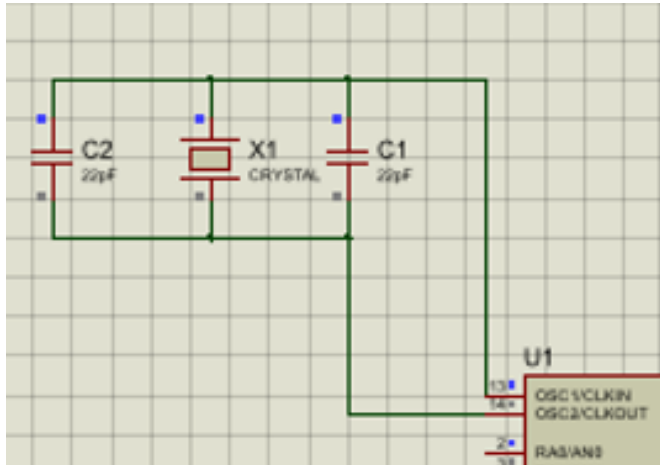


Figure 3: Oscillator clock.

3.3 Input switches

The four button switches (SW1, SW2, SW3 and SW4) are connected to PORTB pins of the microcontroller. Each switch includes a resistor network (R1, R2, R3 and R7) connected to a power source.

- SW1 (Engine Start) - initiates the system
- SW2 (Coolant Confirmation) - validates system conditions
- SW3 (Fan Activation) - triggers motor
- SW4 (Reset) – Reset

These switches simulate automotive sensor inputs such as ignition signals and coolant status. Also, the switches only work when pressed in order as written in the coding.

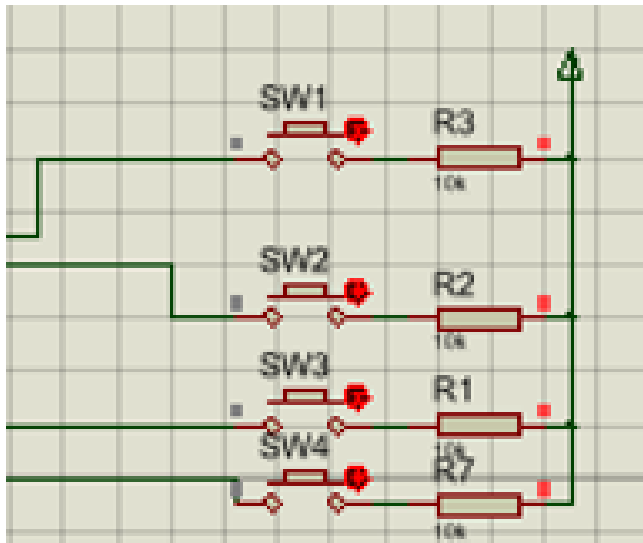


Figure 4: Input switches.

3.4 LEDs

Three LEDs are connected via resistors (R4, R5, R6) to PORTB outputs:

- D1 (Red LED) (Flashes 3 times) – System shut down
- D2 (Green LED) – Engine startup/Coolant OK
- D3 (Yellow LED) - Alert indicator

Each LED provides a clear visual indication of the system status. The resistors 220Ω are used to limit current and protect the LEDs from damage.

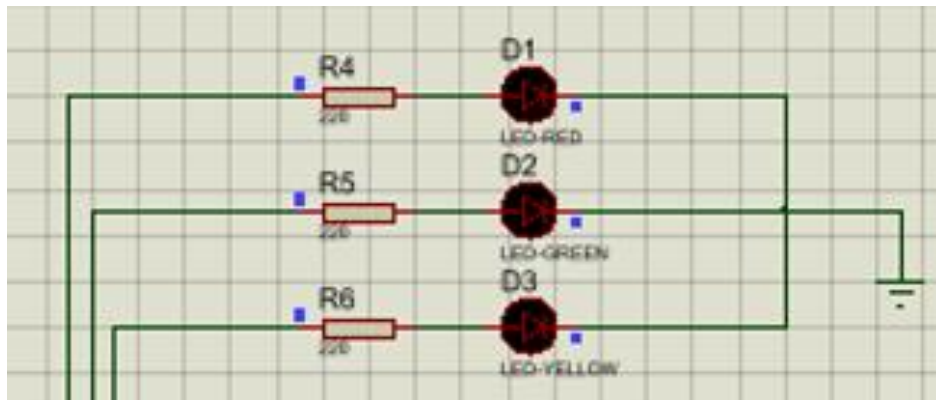


Figure 5: LED system status indicators.

3.5 LCD

The LCD allows embedded systems to provide feedback to the user [5]. A 16×2 LCD module (LCD1) is connected to the microcontroller using PORTD in 4-bit mode. This configuration reduces the number of required I/O pins while maintaining full display functions.

The LCD displays system messages such as:

- “System Ready”
- “Engine Started”
- “Coolant OK”
- “Alert, Fan Activated”
- “System shutdown”
- “Reset”

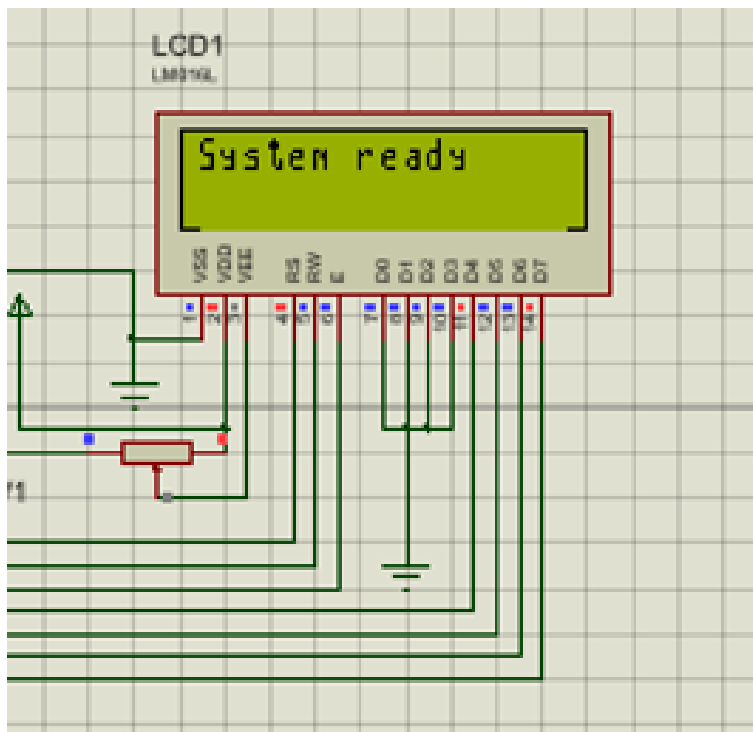


Figure 6: 16×2 LCD module.

3.7 Motor simulation

The DC motor is connected to PORTA pins (RA0 and RA1). These pins control the motor direction:

- IN1 = High, IN2 = Low - Forward rotation
- IN1 = Low, IN2 = Low - Stop

This motor simulates a real world cooling system. Cooling fans are used to generate airflow and remove heat. The fans are typically controlled using variable speed techniques, such as Pulse Width Modulation (PWM). This allows efficient temperature regulation [7].

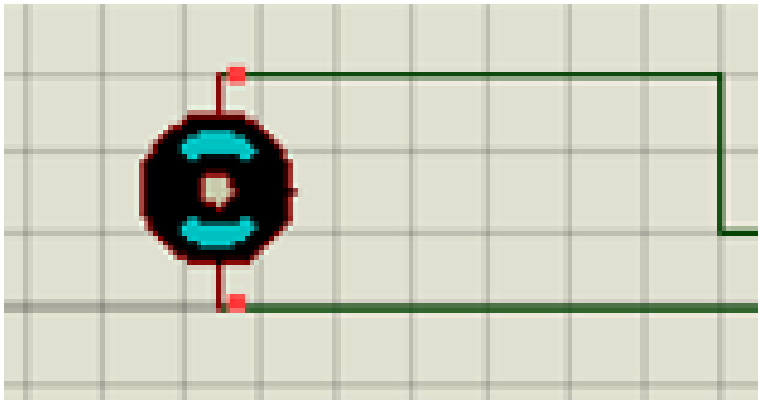


Figure 8: Radiator fan simulator.

4. Program logic

Figure 9, is a flow chart illustrating the control logic of the automotive cooling system. The system progresses through engine start, coolant confirmation, fan activation, motor operation, alert indication and shutdown stages.

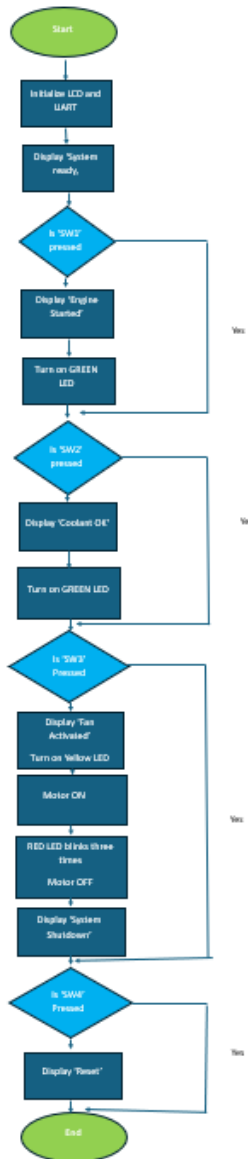


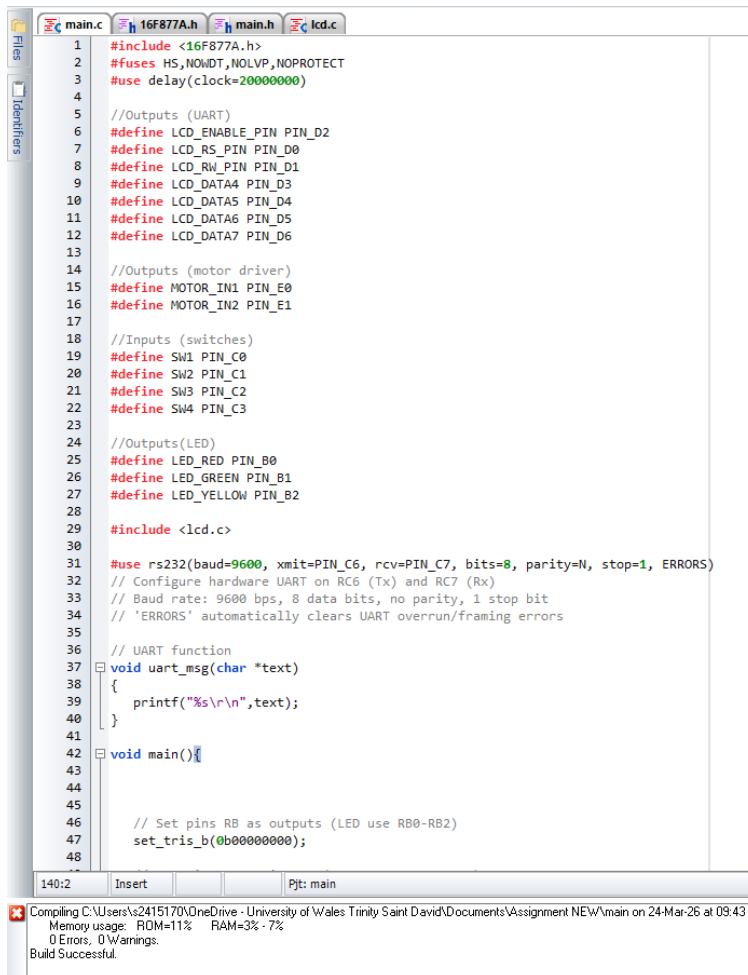
Figure 9: Flow chart of system

4.1 Software implementation

The system was programmed in CCS C, as shown in figures 10,11,12 below, using the PIC16F877A to control the sequence of the automotive cooling system simulator. The microcontroller reads push-button inputs (engine start, coolant check, and fan activation) and processes them only when they are pressed in the correct order.

Outputs are controlled through LEDs for status indication, then a DC motor to simulate the cooling fan and an LCD display for real time messages. UART communication is also implemented at 9600 bps to send system updates to the Proteus virtual terminal.

The program uses conditional logic and timing delays to manage each stage, including a 3-second motor operation and a flashing alert LED before shutdown. The system then waits for a reset input, showing controlled embedded process.



```

1  #include <16F877A.h>
2  #fuses HS,NOWDT,NOLVP,NOPROTECT
3  #use delay(clock=2000000)
4
5  //Outputs (UART)
6  #define LCD_ENABLE_PIN PIN_D2
7  #define LCD_RS_PIN PIN_D0
8  #define LCD_RW_PIN PIN_D1
9  #define LCD_DATA4 PIN_D3
10 #define LCD_DATA5 PIN_D4
11 #define LCD_DATA6 PIN_D5
12 #define LCD_DATA7 PIN_D6
13
14 //Outputs (motor driver)
15 #define MOTOR_IN1 PIN_E0
16 #define MOTOR_IN2 PIN_E1
17
18 //Inputs (switches)
19 #define SW1 PIN_C0
20 #define SW2 PIN_C1
21 #define SW3 PIN_C2
22 #define SW4 PIN_C3
23
24 //Outputs(LED)
25 #define LED_RED PIN_B0
26 #define LED_GREEN PIN_B1
27 #define LED_YELLOW PIN_B2
28
29 #include <lcd.c>
30
31 #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, bits=8, parity=N, stop=1, ERRORS)
32 // Configure hardware UART on RC6 (Tx) and RC7 (Rx)
33 // Baud rate: 9600 bps, 8 data bits, no parity, 1 stop bit
34 // 'ERRORS' automatically clears UART overrun/framing errors
35
36 // UART function
37 void uart_msg(char *text)
38 {
39     printf("%s\r\n",text);
40 }
41
42 void main()
43 {
44
45     // Set pins RB as outputs (LED use RB0-RB2)
46     set_tris_b(0b00000000);
47
48

```

140:2 Insert Pjt: main

Compiling C:\Users\2415170\OneDrive - University of Wales Trinity Saint David\Documents\Assignment NEW\main on 24-Mar-26 at 09:43
Memory usage: ROM=11% RAM=3% - 7%
0 Errors, 0 Warnings.
Build Successful.

Figure 10: CCS Coding.

```

46 // Set pins RB as outputs (LED use RB0-RB2)
47 set_tris_b(0b00000000);
48
49 // Set pins RC as inputs (SW1-SW4 use RC0-RC3)
50 set_tris_c(0b00001111);
51
52 // Set pins RE as outputs (Motor use RE0-RE1)
53 set_tris_e(0b00000010);
54
55 lcd_init();
56
57 output_low(PIN_B0);
58 output_low(PIN_B1);
59 output_low(PIN_B2);
60
61 printf(lcd_putc, "\fSystem ready");
62 printf("System ready");
63
64
65
66 while (!input(SW1));
67 delay_ms(50); // Engine start
68
69
70 output_low(PIN_B0);
71 output_high(PIN_B1); // Turn on the LED
72 output_low(PIN_B2);
73
74 printf(lcd_putc, "\fEngine Started");
75 printf("Engine Started");
76
77 while (!input(SW2));
78 delay_ms(50); // Coolant sensor trigger
79
80
81 output_low(PIN_B0);
82 output_low(PIN_B2);
83 output_high(PIN_B1); // Turn on the LED
84
85 printf(lcd_putc, "\fCoolant OK");
86 printf("Coolant OK");
87
88 while (!input(SW3));
89 delay_ms(50); // wait fan trigger
90
91 output_low(PIN_B0);
92 output_low(PIN_B1);
93 output_high(PIN_B2); // Turn on the LED

```

Figure 11: CCS C coding.

```

92     output_low(PIN_B1);
93     output_high(PIN_B2); // Turn on the LED
94
95
96     printf(lcd_putc, "\fAlert,Fan activated");
97     Printf("Alert, Fan activated");
98     delay_ms(200);
99
100
101     // Motor ON
102     output_high(MOTOR_IN1);
103     output_low(MOTOR_IN2);
104     delay_ms(3000);
105
106
107     for(int i=0;i<3;i++)
108     {
109         output_high(PIN_B0);
110         delay_ms(200);
111         output_low(PIN_B0);
112         delay_ms(200);
113     }
114     output_low(PIN_B2);
115
116     printf(lcd_putc, "\fShutdown Done");
117     Printf("Shutdown Done");
118     delay_ms(200);
119
120
121     // Motor OFF
122     output_low(MOTOR_IN1);
123     output_low(MOTOR_IN2);
124
125
126
127
128
129     while (!input(SW4));
130     delay_ms(50); // Reset
131
132     printf(lcd_putc, "\fReset");
133     Printf("Reset");
134
135
136
137
138     while(TRUE);
139 }

```

Figure 12: CCS C coding.

5. Simulation results

Table 1: Simulation results table.

Test	Input	Expected Result	Actual Result
1	Power ON	System Ready	Pass
2	SW1	Engine Started	Pass
3	SW2	Coolant OK	Pass
4	SW3	Motor Running	Pass
5	3 sec delays	Motor stops	Pass
6	Post motor	Alert, Fans activated	Pass
7	End	Shutdown Done	Pass
8	SW4	Reset	Pass

6. Discussion

This cooling system simulator successfully demonstrates how a microcontroller would control a real world process. The system responds correctly to switch inputs representing engine start, coolant condition, and fan activation and provides clear feedback through LEDs, the LCD, and UART communication.

The software implementation allows that each stage only occurs in the correct order, which reflects real automotive control systems. The motor simulation effectively represents radiator fan operation, and the alert sequence clearly indicates system shutdown.

If further developed, additional features such as a dedicated reset indicator BLUE LED would be added to signal system reset and a fan RPM in place of the motor to see real time running speed.

However, the system uses push button inputs instead of real sensors, and the motor runs at a fixed speed. The use of delay functions also limits efficiency, as the system cannot perform multiple tasks at the same time.

Overall, the system performs well and demonstrates key embedded system concepts effectively.

7. Conclusion

This project successfully designed and simulated an automotive cooling system status simulator using the PIC16F877A microcontroller. The system correctly followed the required sequence of engine start, coolant confirmation, fan activation, motor operation, and shutdown.

The simulation results confirmed that all components operated as expected, with accurate input handling and clear output feedback through LEDs, LCD, and UART communication.

The project demonstrates how embedded systems can be used to control real world processes.

8. References

- [1] M. J and M. J, “Automotive Embedded Systems – Applications, examples, and future trends,” Dec. 03, 2025. <https://www.emertxe.com/blog/2025/12/02/automotive-embedded-systems/>
- [2] A. Makam, “Engine Electrified Cooling System: system and control design in simulation environment,” 2019.
- [3] M. Lab, “PIC16F877A Microcontroller Introduction and features,” *Microcontrollers Lab*, Dec. 02, 2025. <https://microcontrollerslab.com/pic16f877a-introduction-features/>
- [4] “Quartz crystal oscillators,” *Electronics Tutorials*. <https://www.electronics-tutorials.ws/oscillator/crystal.html> (accessed Apr. 12, 2026).
- [5] “LCD Display Module: Enhancing user experience in embedded systems,” *Dongguan Team Source Display Technology Co., Ltd.* <https://www.lcdlcm.com/blog/lcd-display-module-enhancing-user-experience-in-embedded-systems>
- [6] Proculustech, “1-UART TFT LCD Module,” *Proculus Technologies Co., Ltd.*, Apr. 16, 2025. <https://www.proculustech.com/the-essential-role-of-uart-protocols-in-embedded-systems/>
- [7] “Why and how to control fan speed for cooling electronic equipment,” *Analog Dialogue*. <https://www.analog.com/en/resources/analog-dialogue/articles/how-to-control-fan-speed.html> (accessed Apr. 14, 2026).

