

Embedded C TM

by Tom Morris

Submission date: 17-Apr-2026 11:16AM (UTC+0100)

Submission ID: 279981664

File name: 57735_Tom_Morris_Embedded_C_TM_728569_410231666.docx (1.77M)

Word count: 2275

Character count: 11361

**ACDA4013 L4 -
Embedded C -
2025/2026 2nd Semester**

**HID Global Apprentice
Student No: 2413111**

**Electrical & Electronic
Engineering
Degree Apprenticeship**



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Abstract

This document is to present the design and simulation of an automotive cooling system status simulator using a PIC16F877A microcontroller. The design of the schematic and the simulation will be controlled using the Proteus Professional 9 software. The code language being used with the PIC16F877A microcontroller is CCS C and the compiler being used to generate the C source file, the hex file and the microcontroller H file is PCW CCS C Compiler.

It will show, using the tools mentioned above, a logical sequence that manages the engine cooling process using an engine start, coolant monitoring, fan activation and a system shutdown. This is achieved by button-based inputs that will simulate each stage of the sequence once pressed within a specific order. In doing so it will active a specific LED to show what part of the sequence it is in while sending messages to an LCD display and messages to a UART virtual terminal for real-time monitoring.

Table of Contents

Abstract	2
1. Introduction.....	4
2. Background Research	5
2.1 Switches	5
2.2 PIC Microcontroller PIC16F877A	5
2.3 Crystal Oscillator	6
2.4 LCD Module.....	6
2.5 LED and Current Limiting Resistors.....	7
2.6 DC Motor.....	7
2.7 UART Virtual Terminal	8
3. Circuit Design	9
4. Flowchart & Pseudocode	10
4.1 Defining Ports and Pins	10
4.2 SW1 & SW2 Code	13
4.3 SW3 Code	15
4.4 RESET Code.....	17
5. Simulation & Testing.....	19
6. Conclusion	23
7. References	24

1. Introduction

This document shows the different stages of design and simulation of an Automotive Cooling process using a PIC16F877A microcontroller and the Proteus Professional software to create a schematic and run the source code created to simulate the logical sequence of a vehicle engine cooling process.

2. Background Research

2.1 Switches

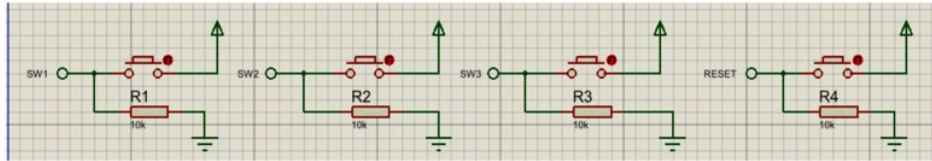


Figure 1 - Switches used in Schematic Design

Figure 1 shows the switches SW1, SW2, SW3 and the RESET switch, each of them are connected to a pull-down resistor. According to Power Electronics [1] pull down resistors are connected between an input pin of microcontroller and to ground. This ensures that the signal line is in a default state of a low logic level of 0. This is so that the input pins don't cause undefined behaviour within the circuit due to input pins having a high impedance and preventing floating inputs.

2.2 PIC Microcontroller PIC16F877A

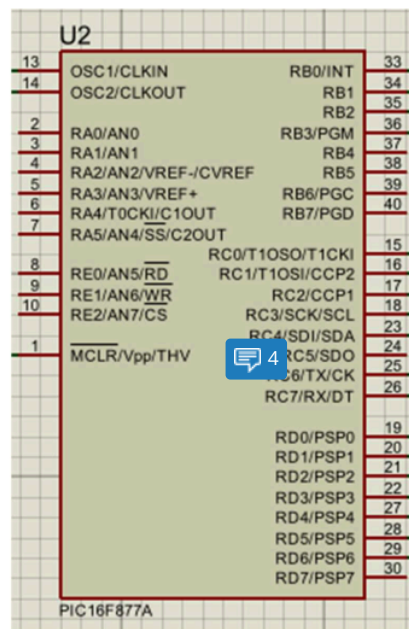


Figure 2 - PIC16F877A Microcontroller

Shown in Figure 2 the PIC16F877A is designed using the RISC architecture and according to Microchip Technology [2] it allows the PIC to operate instructions quickly while using the memory effectively. The PIC16F877A has a 35 single word instruction set and can operate up to a 20MHz frequency, it has 40 pins and 33 of those pins are

able to be configured and runs between 4.2V to 5.5V. It also has UART for serial communication and also has SPI and I2C for synchronous serial communication.

2.3 Crystal Oscillator

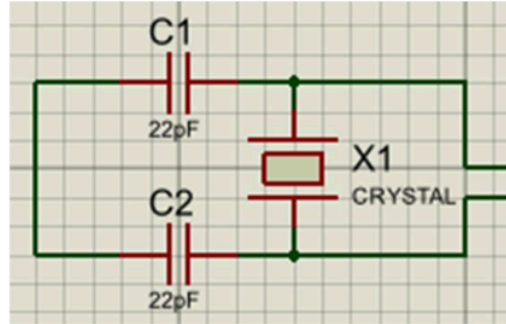


Figure 3 - Crystal Oscillator

Shown in Figure 3 is a Crystal Oscillator. According to Electricity-Magnetism [3], a crystal oscillator provides a clock signal to the microcontroller, this is a rhythmic pulse that is provided to the circuit to enable the synchronisation of the circuit and provides the timing signals needed to run the instructions provided to the circuit.

2.4 LCD Module

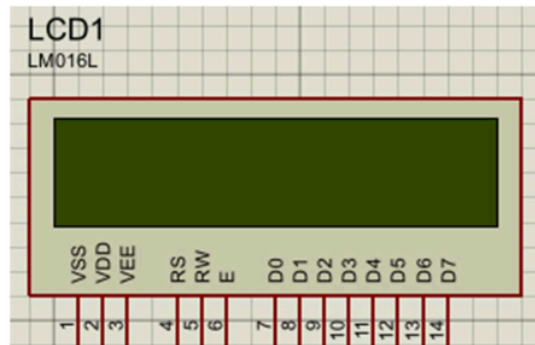


Figure 4 - LCD Module LM016L

Shown in Figure 4 is an LCD Module, the LCD Module being used within the circuit is a 16x2 LCD which is used to display alphanumeric information. According to [4] 16x2 shows that the LCD is able to display across 16 columns and 2 rows, it also has 14 pins. This allows the LCD module to display 32 characters at one time.

2.5 LED and Current Limiting Resistors

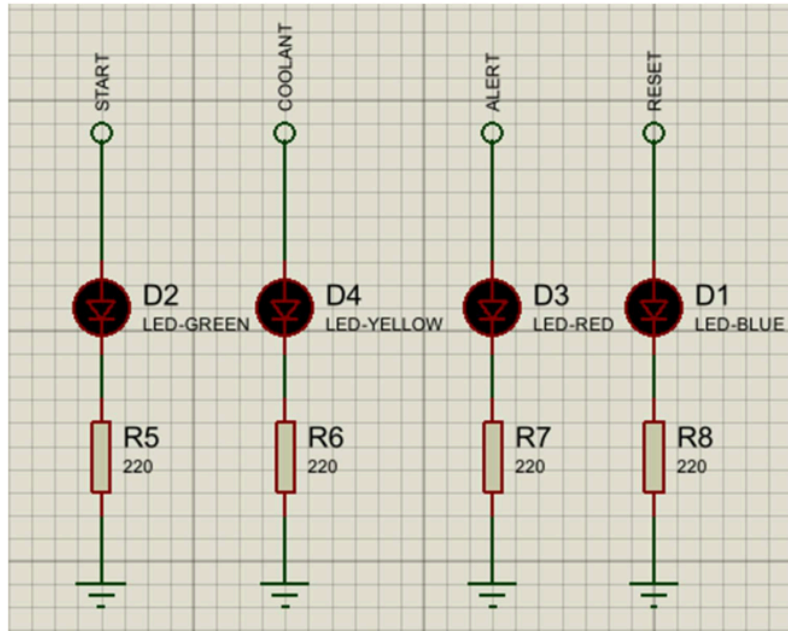


Figure 5 - LEDs with Current Limiting Resistors

Shown in Figure 5 are current limiting resistors. According to Adafruit [5] LEDs are used as they have a low power consumption, a longer lifespan and are fast switching.

According to Waveform Lighting [6], LEDs are current driven and even a slight increase in voltage could cause potential damage through overheating. The resistor acts to keep the LED within its safe operating range by stopping any potential surges that may happen while the LED is operating.

2.6 DC Motor



Figure 6 - DC Motor

Shown in Figure 6 is a DC motor which is being used as a fan, according to [7] the advantages of having a DC motor as a fan is that they have a high energy efficiency, they are able to convert more of the energy they consume into useable energy. They are used to help dissipate heat that is generated by electrical components which in turn extends

the lifespan of the components being used within the system while enhancing the overall performance of the system.

2.7 UART Virtual Terminal

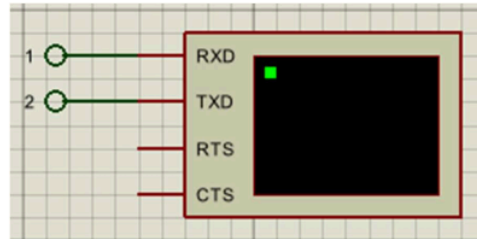


Figure 7 - UART Virtual Terminal

UART is a hardware device that is used for serial communication between two devices, according to [8] they are able to monitor device behaviour and send or receive data which is useful when debugging devices and programming while needing little hardware to operate. To transmit and receive data, the UART has two wires, one is TX which is used for transmitting data and the other is RX which is used to receive data from another device. Both devices need to be operating at the same baud rate.

3. Circuit Design

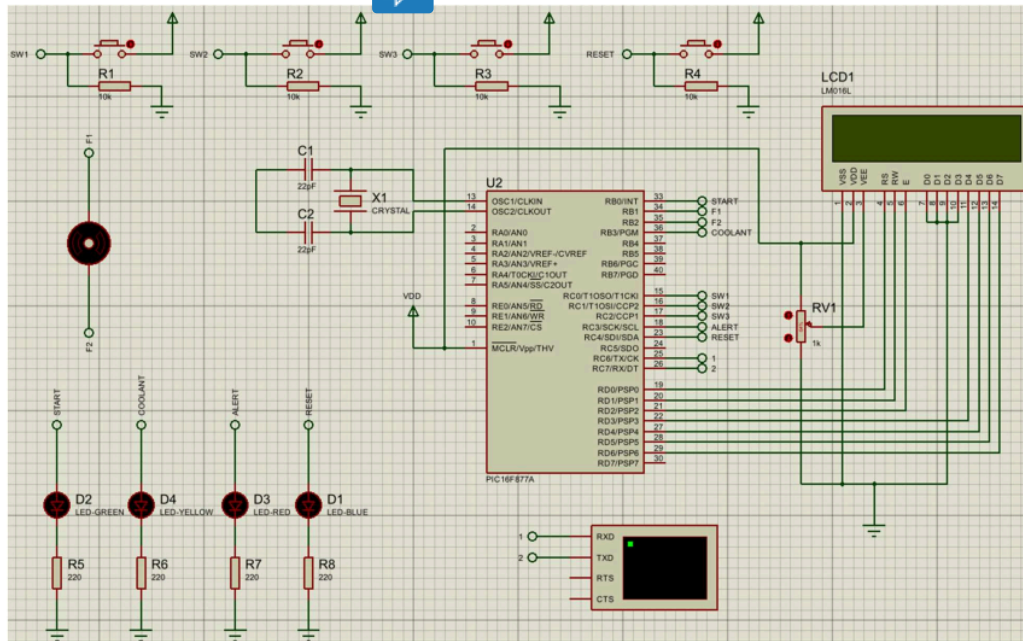


Figure 8 - Completed Circuit Design

Shown in Figure 8, is the completed circuit design. Each switch is connected to a 10k pull down resistor to ensure a low logic (0) when inactive these control each stage of the simulation (SW1, SW2, SW3, RESET).

The PIC16F877A is used to read the 4 switch inputs and then drives the 4 different LEDs depending on what button is pressed. It also sends system messages to the UART virtual terminal and controls what the LCD displays and activates the DC motor.

The crystal oscillator is connected to the OSC1 and OSC2 pins and is a 4MHz crystal with the 22pF capacitors.

The LCD module is used to display system messages. The VSS pin and the D0-D3 pins are grounded to enable the data lines to be in 4-bit mode with D4-D7 being connected to RD3-RD6. The control signals RS, RW and E are connected to RD0-RD2. RV1 is a pot connected to VDD (power) and VEE to control the contrast of the LCD.

The DC motor is controlled by the two RB1 and RB2 pins on the PIC16F877A which turns it on during the cooling process and turns off afterwards.

The UART terminal is used to display status messages through the PIC16F877A pins RC6 and RC7 to the RX and TX pins on the UART virtual terminal, this enables the circuit to have real time system monitoring.

4. Flowchart & Pseudocode

4.1 Defining Ports and Pins

```
#include <Coursework.h>
#define LCD_ENABLE_PIN PIN_D2
#define LCD_RS_PIN PIN_D0
#define LCD_RW_IN_D1
#define LCD_DATA_IN_D3
#define LCD_DATA5 PIN_D4
#define LCD_DATA6 PIN_D5
#define LCD_DATA7 PIN_D6

#include <lcd.c>

#define START PIN_B0
#define F1 PIN_B1
#define F2 PIN_B2
#define COOLANT PIN_B3
#define SW1 PIN_C0
#define SW2 PIN_C1
#define SW3 PIN_C2
#define ALERT PIN_C3
#define RESET PIN_C4
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, bits=8, parity=N, stop=1)
```

Figure 9 – CCS C Code Defining Pins

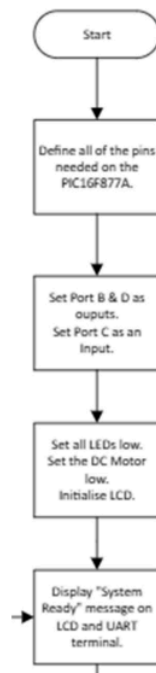


Figure 10 - Flowchart of Defining Pins

Shown in Figure 9, is the first stage of code with all the defined pins that are going to be used throughout the simulation. The LCD pins were defined based on how it was wired on the schematic, from D0-D7 defining the enable pin as D2.

The pins used for the switches, motor, LEDs and UART are defined using the same naming conventions used on the schematic. To use the UART the baud rate, number of bits, pins and parity need to be defined for the RS232.

```
void main()
{
    set_tris_b(0x00);           // Set PORTB as output
    set_tris_c(0xFF);           // Set PORTC as input
    set_tris_d(0x00);           // Set PORTD as output
    output_low(START);           // Set Start as low
    output_low(COOLANT);         // Set Coolant as low
    output_low(ALERT);           // Set Alert as low
    output_low(RESET);           // Set Reset as low
    output_low(F1);              // Set F1 as low
    output_low(F2);              // Set F2 as low
    lcd_init();                  // Initialise the LCD
    printf("System Ready\r\n");  // System Ready message on UART Virtual Terminal
    lcd_putc("\fSystem Ready");  // System Ready message on LCD
}
```

Figure 11 - Initial States for Components

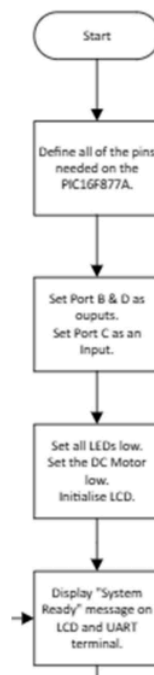


Figure 12 - Configuring Ports and Components Flowchart

Shown in Figure 11, port B and D were set as outputs and port C was set as an input. The LCD was initialised to enable messages to be sent and the first set of messages to show

the system is ready to run were placed here as well. The different LEDs and other pins were all set as low as well this gives these ports and pins a known starting point when the simulation is running.

4.2 SW1 & SW2 Code

```
while(TRUE)
{
    while(input(SW1))
    {
        output_high(START);           // Turn the Green LED ON
        lcd_putc("\fEngine Started"); // Show Engine Started message on LCD
        delay_ms(200);                // Delay
        printf("Engine Started\r\n"); // Send Engine Started message to the terminal
    }
    while(input(SW2))
    {
        output_low(START);           // Turn the Green LED OFF
        output_high(COOLANT);        // Turn the Yellow LED ON
        lcd_putc("\fCOOLANT OK\r\n"); // Show Coolant OK message on LCD
        delay_ms(200);                // Delay
        printf("COOLANT OK\r\n");     // Send Engine Started message to the terminal
    }
    while(input(SW3))
}
```

Figure 13 - SW1 & SW2 Code

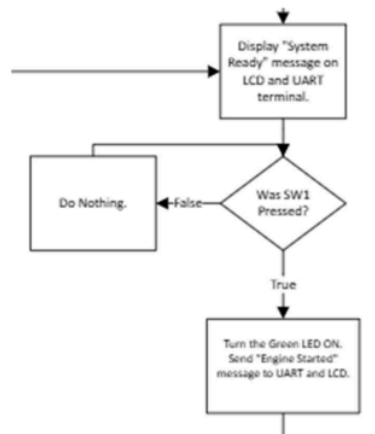


Figure 14 - SW1 Flowchart

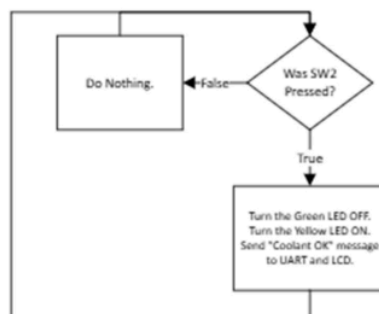


Figure 15 - SW2 Flowchart

Shown in Figure 13, is the while true function that creates an infinite loop. For the engine started process, the “while(input(SW1))” command will only run if the button is pressed,

the “output_high(START)” is to drive the green LED high (on) and then send the message “Engine Started” to both the LCD and UART terminal while introducing a delay in between to reduce bouncing and producing an echo of the same message.

The same is shown for “while(input(SW2))” command will only run if the button is pressed otherwise it will stay in the previous state. For the coolant check process, the “output_low(START)” is to drive the green led to a low (off) state. The “output_high(COOLANT)” is to drive the yellow LED high (on) and then send the message “COOLANT OK” to both the LCD and UART terminal to replace the previous message while introducing a delay in between to reduce bouncing and producing an echo of the same message.

4.3 SW3 Code

```
while(input(SW3))
{
    output_low(COOLANT);           // Turn Yellow LED OFF
    output_high(F1);               // Initialise motor output HIGH (motor on)
    output_low(F2);               // Initialise motor output LOW (motor on)
    lcd_puts("\fMotor Running\nFan Activated"); // Show Motor Running message on LCD
    printf("Motor Running... Fan Activated\r\n"); // Send Motor Running message to the terminal

    delay_ms(3000);               // Delay

    output_high(ALERT);           // Flash Red LED ON
    delay_ms(300);               // Delay
    output_low(ALERT);           // Flash Red LED OFF
    delay_ms(100);               // Delay
    output_high(ALERT);          // Flash Red LED ON
    delay_ms(300);               // Delay
    output_low(ALERT);           // Flash Red LED OFF
    delay_ms(100);               // Delay
    output_high(ALERT);          // Flash Red LED ON
    delay_ms(300);               // Delay
    output_low(ALERT);           // Turn Red LED OFF
    output_low(F1);               // Initialise motor output LOW (motor off)
    output_low(F2);               // Initialise motor output LOW (motor off)
    lcd_puts("\fALERT");          // Show ALERT message on LCD
    printf("ALERT\r\n");          // Send ALERT message to the terminal

    delay_ms(1000);

    lcd_puts("\fSHUT DOWN DONE"); // Show Shut Down Done message on LCD
    printf("SHUT DOWN DONE\r\n"); // Send Shut Down Done message to the terminal
}
```

Figure 16 - SW3 Code

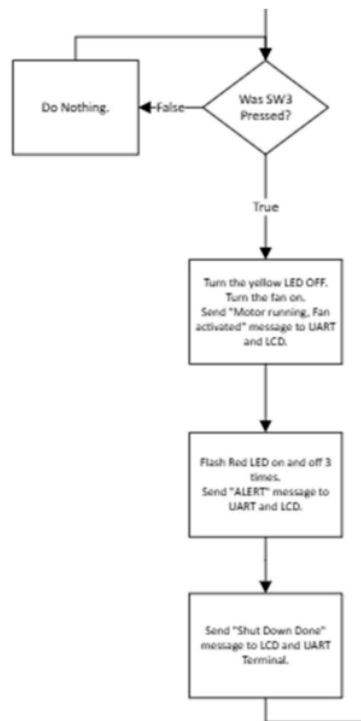


Figure 17 - SW3 Flowchart

Shown in Figure 16 the “while(input(SW3))” command will only run if the button is pressed otherwise it will stay in the previous state. For the Alert stage of the simulation, the “output_low(COOLANT)” is to drive the yellow led to a low (off) state. The “output_high(F1)” is to drive the DC motor high (on) while the “output_low(F2)” keeps the F2 side low and then send the message “Motor Running, Fan Activated” to both the LCD and UART terminal to replace the previous message while introducing a delay in between to reduce bouncing and producing an echo of the same message.

The “output_high(ALERT)” , “delay_ms(300)” and “output_low(ALERT)” create a flashing red led on and off three times to show the alert stage while sending the message “ALERT” to the LCD and UART terminal to replace the previous message. The “output_low(F1)” and “output_low(F2)” then turn the motor off to finish the cooling process and then display the message “Shut Down Done” on the LCD and UART terminal.

4.4 RESET Code

```

while(input(RESET))
{
    output_low(ALERT);           // Turn Red LED OFF
    output_high(RESET);         // Turn Blue LED ON
    lcd_puts("\fRESET COMPLETE\n"); // Show Reset Complete message on LCD
    printf("RESET COMPLETE\r\n"); // Send Reset Complete message to the terminal
    delay_ms(2000);             // Delay
    output_low(RESET);          // Turn Blue LED OFF
    delay_ms(1000);             // Delay
    printf("System Ready\r\n");  // Show System Ready message on LCD
    lcd_puts("\fSystem Ready"); // Send System Ready message to the terminal
}
}

```

Figure 18 - Reset Code

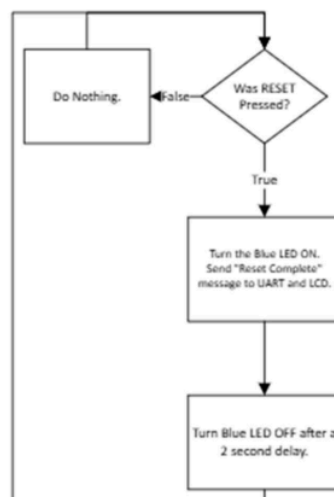


Figure 19 - RESET Flowchart

Shown in Figure 18 the “while(input(RESET))” enables the command to only run when that exact button is pressed or it remains in its previous state. This was added to the code to enable a continuous loop within the simulation.

For the reset stage, the ALERT led is driven low to ensure it is in a known state by the command “output_low(ALERT)”. The blue led is then driven high (on) by the command “output_high(RESET)” and then sends a message to the LCD and UART terminal to show “RESET COMPLETE” and then introduces a delay and then runs the command “output_low(RESET)” to turn the blue led off after 2 seconds. It then returns to the original state of “System Ready” and shows this on the LCD and UART terminal to enable the process to be run again in a loop.

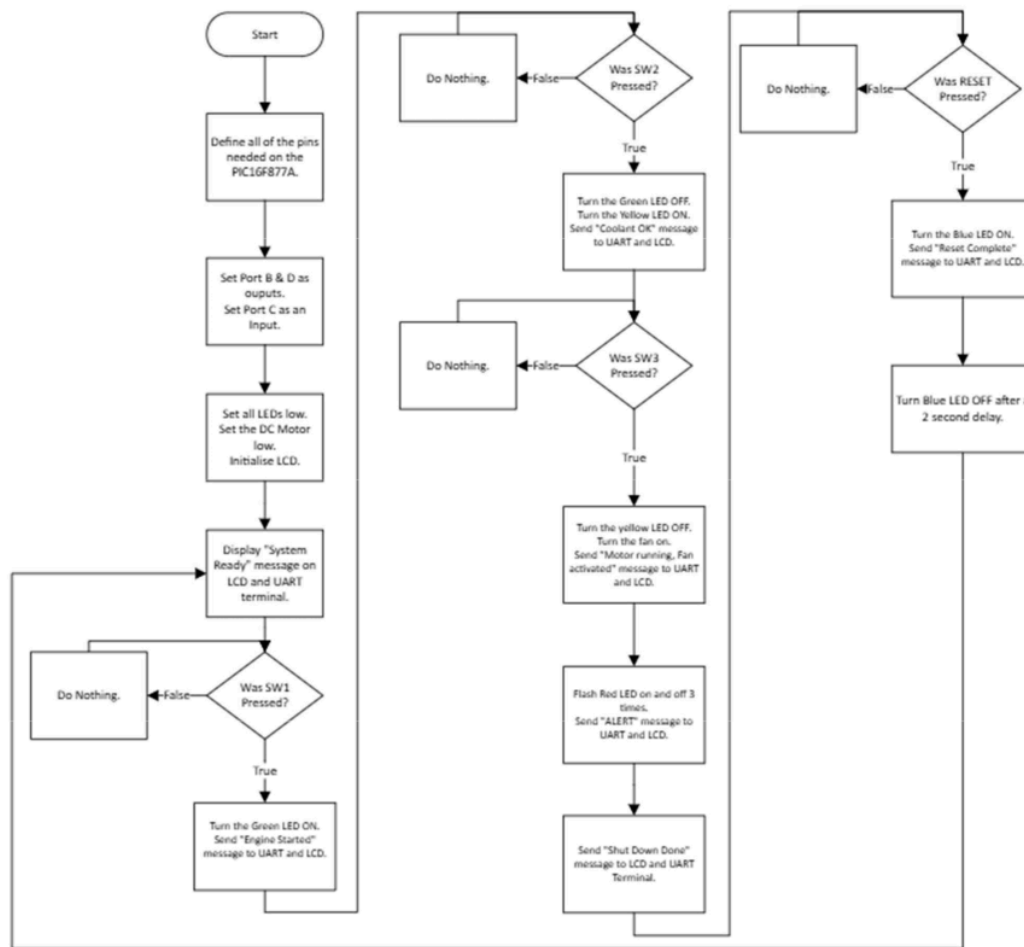


Figure 20 - Completed Flowchart

5. Simulation & Testing

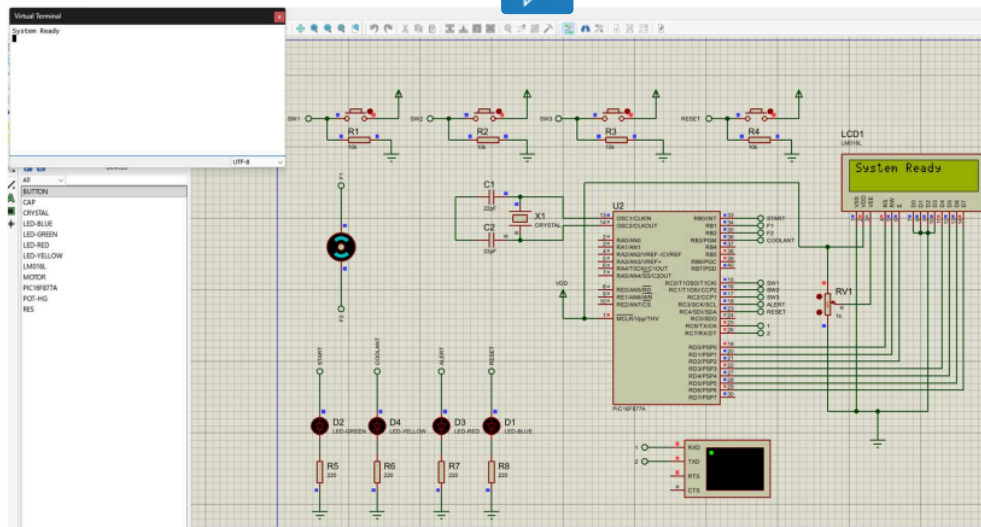


Figure 21 - System Ready Stage

Shown in Figure 21 is the initial stage before any buttons have been pressed and the system is ready to be used.

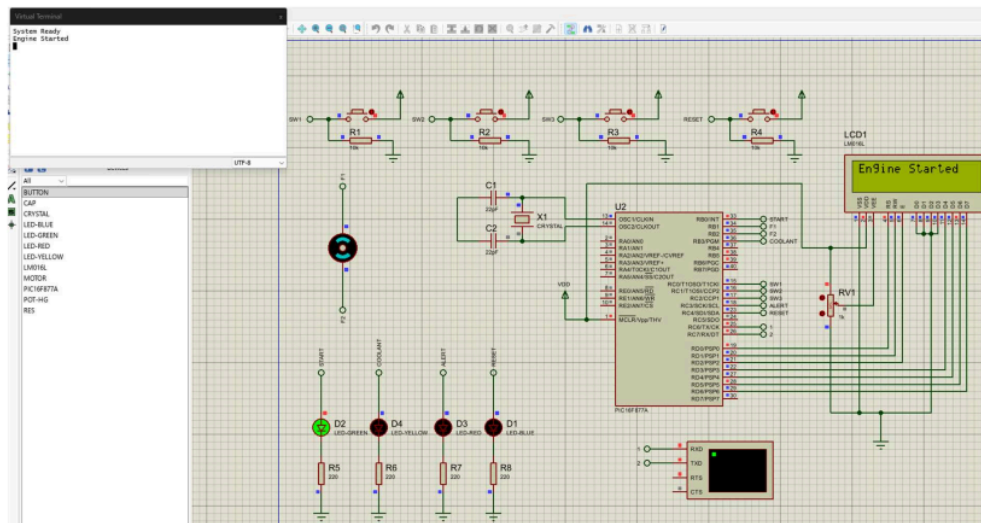


Figure 22 - Engine Started Stage

Shown in Figure 22 is the engine started state showing the displayed lcd and UART messages and the green led on after SW1 was pressed.

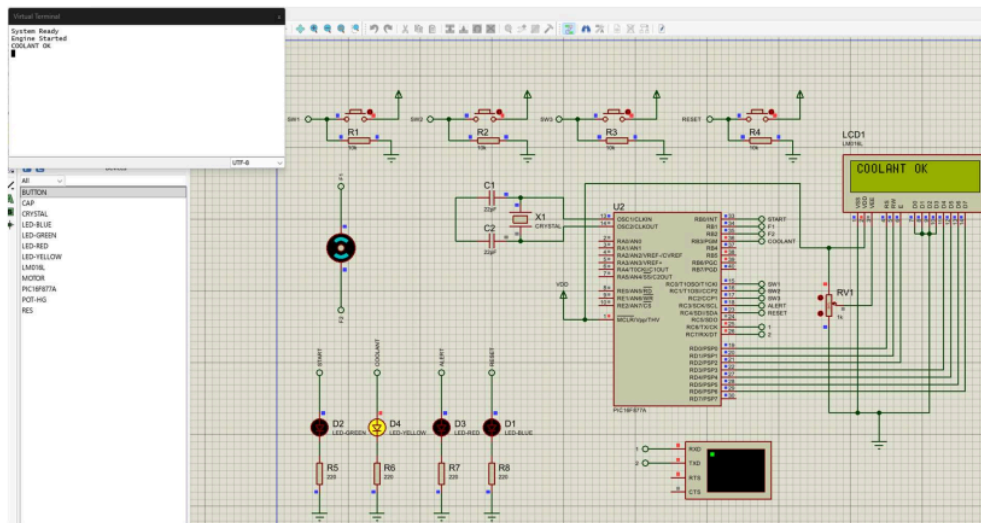


Figure 23 - Coolant Check Stage

Shown in Figure 23 is the coolant check showing the displayed message on the LCD and UART terminal while having the yellow LED on after SW2 was pressed.

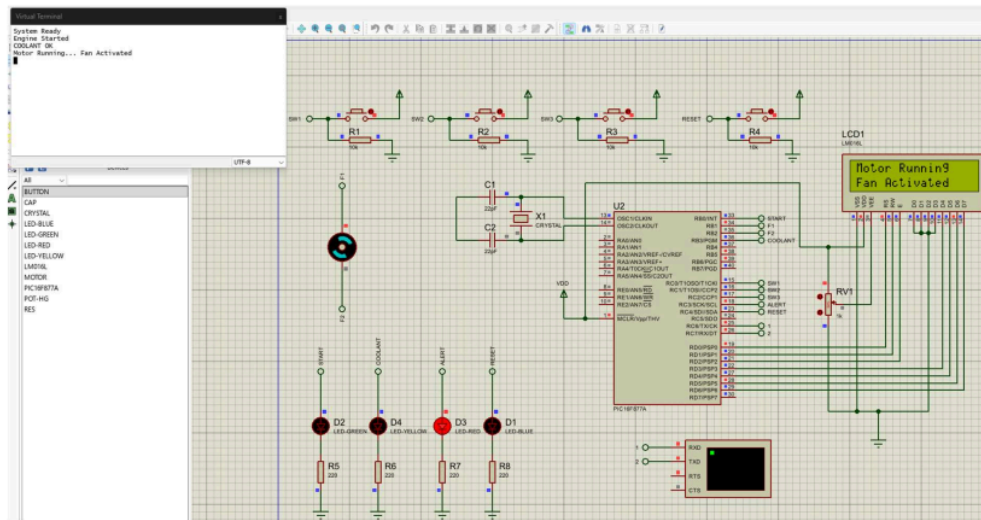


Figure 24 - Alert Stage

Shown in Figure 24 is the alert stage where it flashes the red led 3 times and displays an alert and fan activated message on the LCD and UART and then activates the fan to cool down the system after SW3 was pressed.

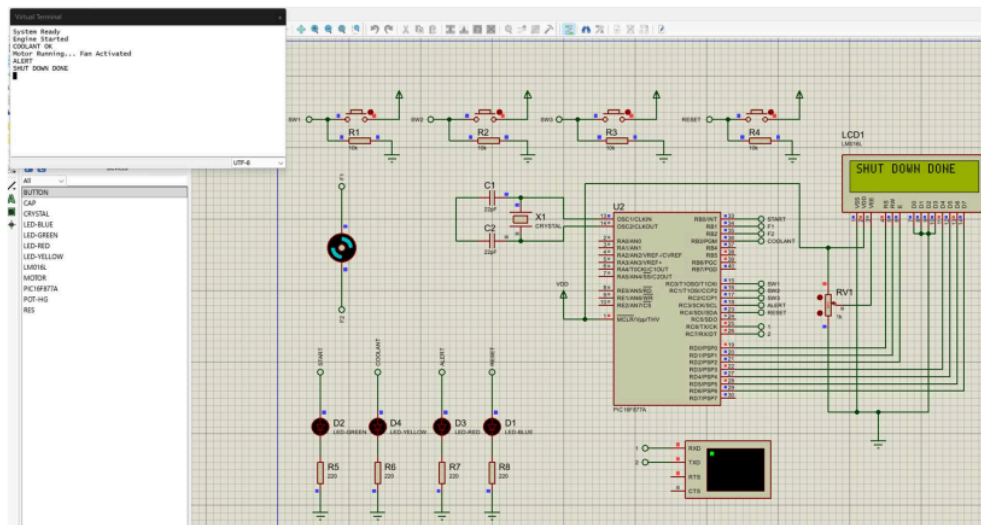


Figure 25 - Shut Down Stage

Shown in Figure 25 is the shutdown stage after the alert and fan have run it shuts the system down and displays the messaged on the LCD and UART terminal.

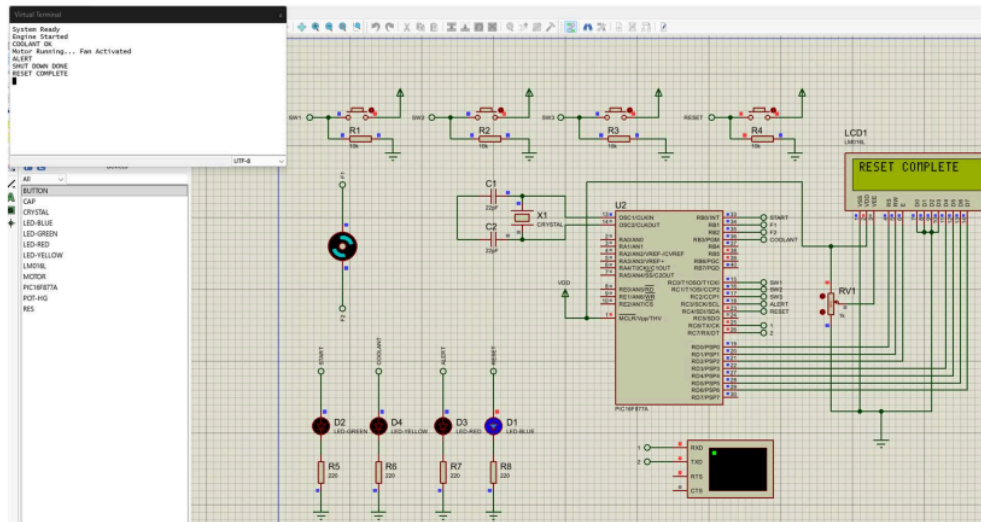


Figure 26 - Reset Stage

Shown in Figure 26 is the reset stage which turns on the blue led and displays the reset complete message on the LCD and UART terminal once complete.

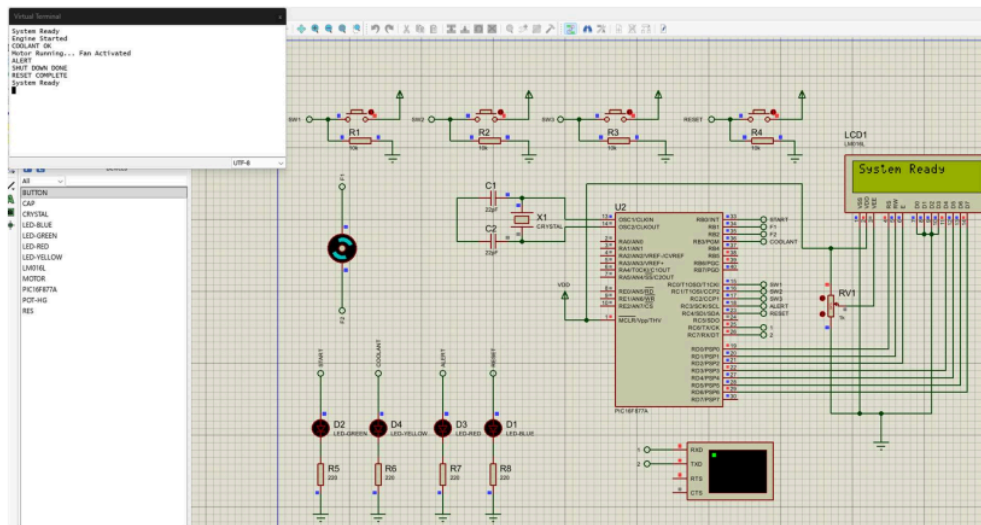


Figure 27 - System Ready Stage

Shown in Figure 27 is the system then reverting back to the first stage to enable the loop to continue when the first switch is pressed again.

6. Conclusion

To conclude the simulation was a success and the code was able to replicate the cooling process of an engine management system and produce a correct representation of each stage and added in a reset to enable the system to run the operational sequence correctly in an infinite loop while providing LCD and UART real time monitoring. This shows that the simulation met all project objectives and took a step further by adding a reset within the simulation.



7. References

- [1] M. D. P. Emilio, "Power Electronics News," 18 May 2025. [Online]. Available: <https://www.powerelectronicsnews.com/understanding-pull-down-resistors-a-comprehensive-tutorial/>.
- [2] "Microchip," Microchip Technology, 29 November 2012. [Online]. Available: <https://www.microchip.com/en-us/product/PIC16F877A>.
- [3] "Electricity-Magnetism," Electricity-Magnetism, 26 October 2023. [Online]. Available: <https://www.electricity-magnetism.org/crystal-oscillators/>.
- [4] E. Efyian, "Electronicsforu," 25 August 2023. [Online]. Available: <https://www.electronicsforu.com/technology-trends/learn-electronics/16x2-lcd-pinout-diagram>.
- [5] T. Cooper, "Adafruit," 11 February 2013. [Online]. Available: <https://learn.adafruit.com/all-about-leds/what-are-leds-used-for>. [Accessed 18 July 2025].
- [6] "Waveform Lighting," [Online]. Available: <https://www.waveformlighting.com/pcb-designs/when-and-why-do-leds-need-current-limiting-resistors>.
- [7] "Blauberg Motoren," 22 August 2024. [Online]. Available: <https://www.blaubergmotors.com/news/what-is-a-dc-fan-motor-a-comprehensive-guide-for-industrial-applications/>.
- [8] "Microcontrollerslab," [Online]. Available: <https://microcontrollerslab.com/uart-communication-working-applications/>.



61 /100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5



Comment 1 | Engagement with Literature(10)

The literature review should include points related to the design and provide a critical comparison with similar designs discussed in journal papers and other relevant sources.



Comment 2 | Engagement with Literature(10)

It is essential to explain how the cooling system generally works and how it has been implemented in your design.



Comment 3 | Knowledge & Understanding(20)

Detailed explanations of the I/O components are provided, which is appreciable.



Comment 4 | Cognitive & Intellectual (25)

Some features of the PIC microcontroller are mentioned; however, the justification for selecting the components, especially the PIC microcontroller, should be explained in more detail. Features such as low cost and low power consumption should also be included.

PAGE 6

PAGE 7

PAGE 8

PAGE 9



Comment 5 | Knowledge & Understanding(20)

The circuit design and construction are clearly explained, which is good.

PAGE 10



Comment 6 | Cognitive & Intellectual (25)

The flowchart is included, which is good. However, the pseudocode is not appropriate as it mainly reflects the actual code.

PAGE 11



Comment 7 | Knowledge & Understanding(20)

The code explanations provided are useful, but they should also discuss CCS C constructs, including the program structure, I/O definitions, control statements, I/O configuration, and functions used to give a clearer understanding of how the program is structured.

PAGE 12

PAGE 13



Comment 8 | Cognitive & Intellectual (25)

The LED sequence is broadly explained; however, more detailed discussion is still needed.

PAGE 14

PAGE 15

PAGE 16

PAGE 17



Comment 9 | **Transferable Skills (15)**

The code structure is neat and clear, and the inclusion of comments is appreciated.

PAGE 18

PAGE 19



Comment 10 | **Cognitive & Intellectual (25)**

Testing results are included, which is good. However, more points should be added in the evaluation or reflection section regarding the challenges faced during the design process.



Comment 11 | **Practical Skills (30)**

Practical Observations The design initially works as expected, but the switch and LED sequence is not implemented correctly, as the switches do not operate in the intended order. When a switch is pressed, the UART displays the message twice instead of once, which is not expected. The sequence involving SW3, the motor operation, and the alert LED functions as intended. However, the intermediate switches affect the system in an unintended sequence. There is a reset button, but it has no effect since the switches are not functioning in a proper sequence. The motor is powered at 12V instead of 5V, causing it to run slowly. The circuit is neat and clear, with all component values correctly specified except for the motor, which should be rated at 5V.

PAGE 20

PAGE 21

PAGE 22

PAGE 23



Comment 12 | **Transferable Skills (15)**

A good structure is maintained throughout the report. It could be improved further by adding a critical review of the design, details on the selection of the microcontroller, CCS C structures, and the cooling system concept.

PAGE 24



Comment 13



Engagement with Literature(10)

The referencing is satisfactory; however, it should be consistently maintained throughout the report.

GRADING FORM: EMBEDDED C

TOM MORRIS

61.20

ENGAGEMENT WITH LITERATURE(10)

Research supports embedded automotive cooling logic and PIC-based simulation. Uses scholarly/technical sources and correct referencing.



Adequate sources; mostly descriptive; limited critical evaluation and linkage to design.

6

KNOWLEDGE & UNDERSTANDING(20)

Demonstrates knowledge of sequencing logic, timers/delays, I/O interfacing, LCD 4-bit mode, UART monitoring and real-world cooling workflow.



Satisfactory understanding; basic explanations provided but limited depth in some areas.

12.2

COGNITIVE & INTELLECTUAL (25)

Evaluates design approach and makes justified decisions using logical reasoning. Demonstrates structured thinking and reflective judgement.



Limited analysis; design choices not justified

11.5

PRACTICAL SKILLS (30)

Implements and tests a PIC16F877A Proteus simulation meeting : switch sequencing, LED states, fan motor control, alert blink, LCD and UART messaging.



Basic functionality achieved; some outputs or timing sequencing inconsistent

19

TRANSFERABLE SKILLS (15)

Communicates work clearly and professionally through a structured report and well-documented code. Submits required files and evidence reliably.



Well-structured report; clear communication; good code readability and presentation.

12.5