Microprocessor Interfacing & Programming

EL-3002

Project Report

Fall 2025

Peltier Based Beverage Cooler with ATmega32

Group Members
Muhammad Ibrahim (23i – 6090)
Mohammad Ahtesam (23i – 6060)

Section (A)

Date of submission
05/12/2025

Course Instructor
Dr. Waseem Ikram

Lab Instructor
Engr. Maryam Zafar

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), ISLAMABAD

Abstract

This project implements a beverage cooling system that uses a thermoelectric Peltier module controlled by an ATmega32 microcontroller. Temperature is sensed using an LM35 analog sensor, digitized through the microcontroller's ADC, and displayed on a 16×2 LCD. A 5V relay module, driven by the ATmega32, is used to switch the 240V AC mains supply that powers the 12V adapter for the Peltier module and the cooling fan. When the measured temperature crosses a defined threshold, the microcontroller activates the relay, which energizes the 12V power supply and turns on the cooling assembly.

The system combines thermal conduction components such as aluminium heat spreaders, a steel crucible, a heatsink, thermal paste, and fan-based heat rejection. It demonstrates practical thermoelectric cooling, microcontroller-based sensing, relay-driven control of high-voltage supplies, and real-time monitoring through an LCD interface.

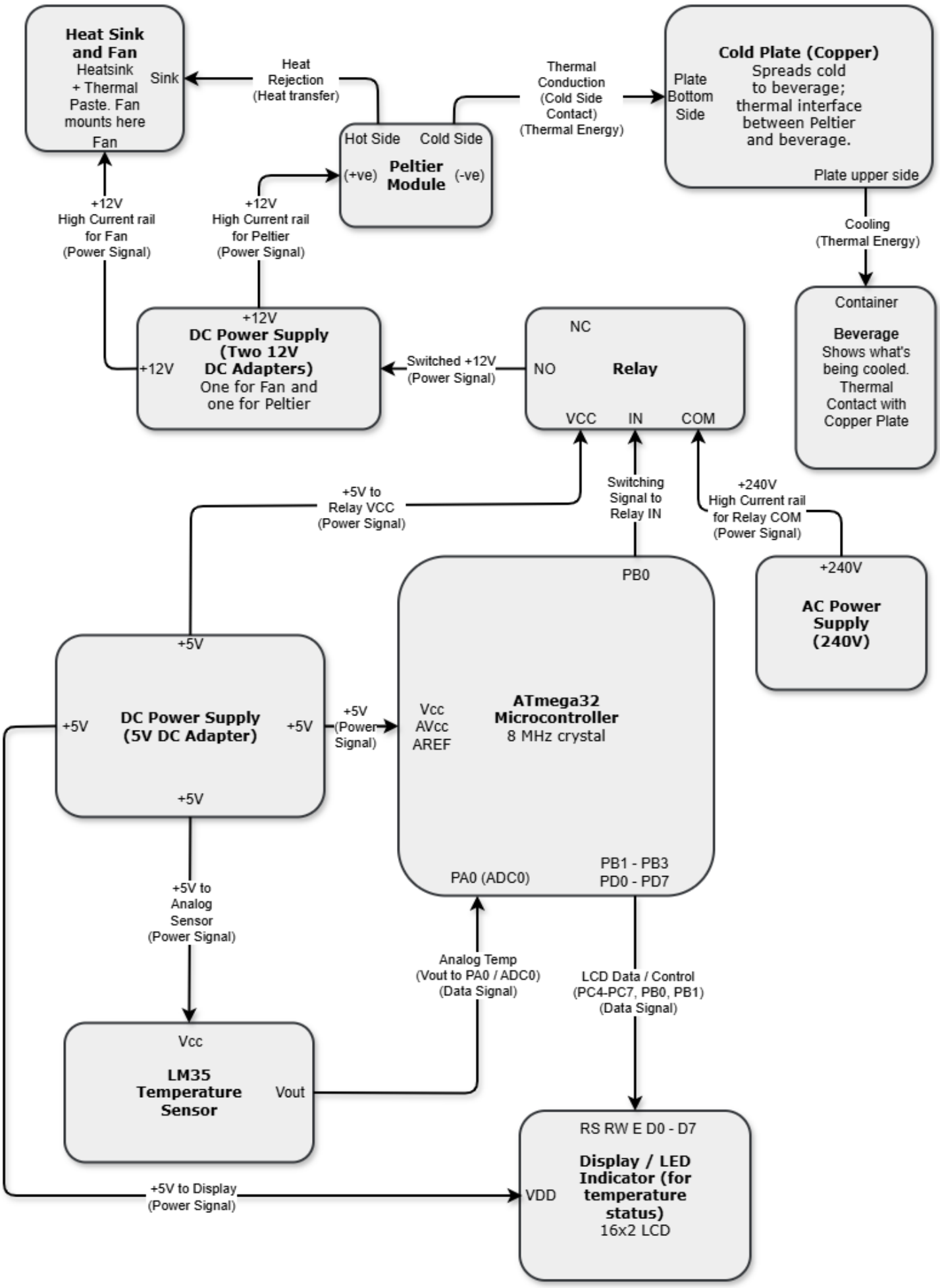
Objectives

1. Build a beverage cooling system using a Peltier TEC1-12706 module.
2. Measure temperature accurately using an LM35 analog sensor with ATmega32 ADC.
3. Display real-time temperature readings on a 16×2 LCD.
4. Use a 5V relay to switch the 240V AC mains powering the 12V DC adapter for the Peltier and fan.
5. Integrate mechanical cooling elements such as heatsinks, aluminium spreaders, and a steel crucible.
6. Analyse cooling performance and control response under real operating conditions.

Components Mapping

Component	Microcontroller Pin	Power Source	Function
LM35 Temperature Sensor	ADC0	5V DC	Temperature measurement
5V Relay Module	PB0	5V DC	Switch AC supply to Peltier & Fan
Peltier TEC1-12706	Relay Output	12V DC Adapter	Cooling the beverage
12V DC Fan	Relay Output	12V DC Adapter	Heat dissipation for Peltier
16×2 LCD	PB1 – PB3 / PD0 – PD7	5V DC	Display temperature readings

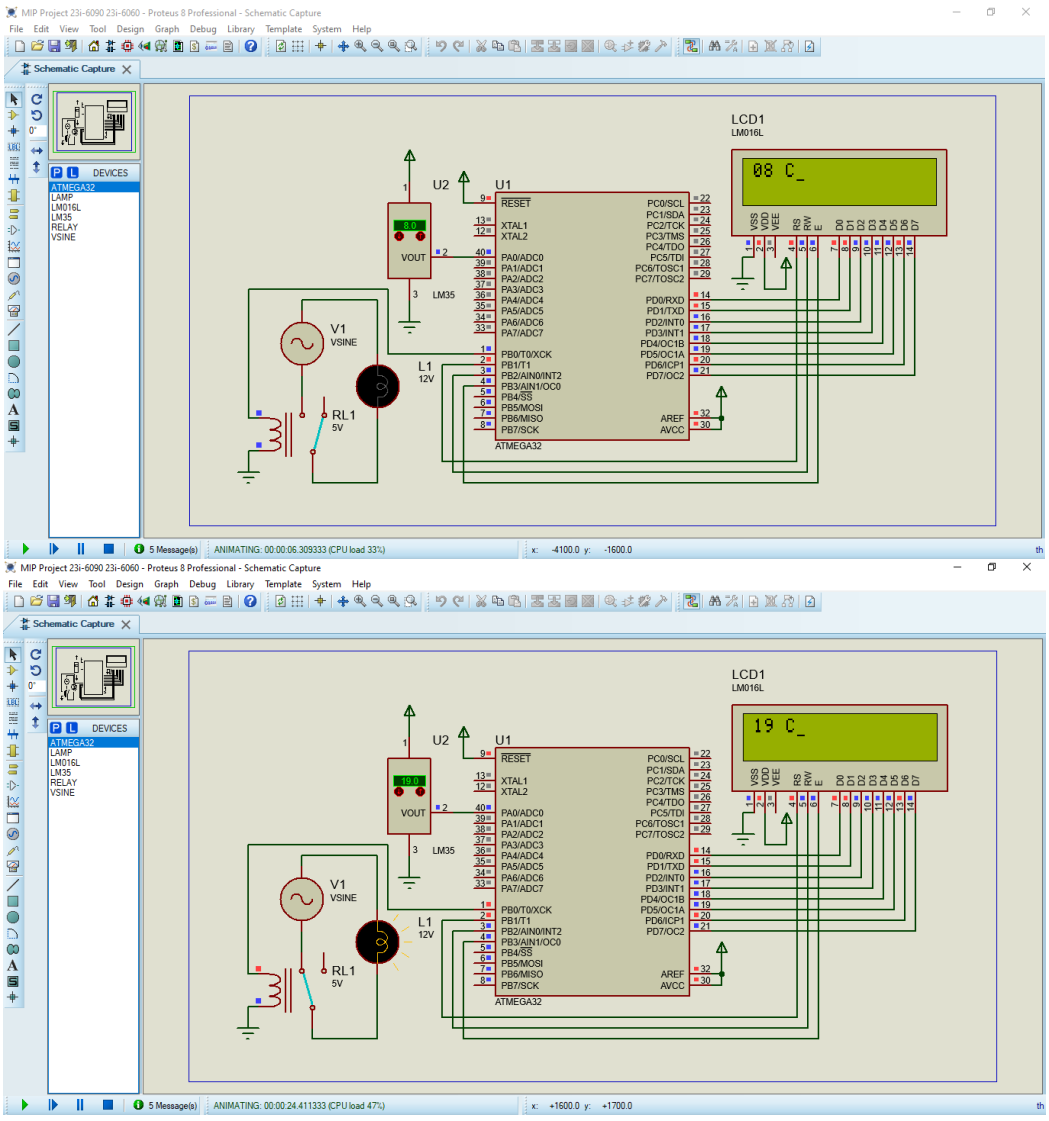
Block Diagram



Working Principle

1. The LM35 sensor continuously measures the beverage temperature and sends analog readings to the ATmega32 ADC.
2. The microcontroller compares the measured temperature to a predefined threshold.
3. If the temperature exceeds the set limit, PB0 outputs a high signal, energizing the 5V relay.
4. The relay switches the 240V AC supply, powering the 12V DC adapter, which in turn powers the Peltier module and the 12V DC fan.
5. Heat generated on the hot side of the Peltier is dissipated by the heatsink and fan, while the cold side cools the beverage.
6. Real-time temperature feedback is displayed on a 16×2 LCD, allowing monitoring and adjustments if needed.

Proteus Simulation: (Here a Lamp is used instead of AC Fan as an example)



Wiring Explanation

1. **Microcontroller to Relay:**
    - PB0 outputs logic HIGH/LOW to the IN pin of the relay.
    - Relay VCC connected to 5V, GND connected to microcontroller GND.
2. **Relay AC Switching:**
    - COM pin of the relay connected to 240V AC mains.
    - Normally Open (NO) pin connected to 12V DC adapter input.
    - When relay energizes, AC powers the adapter, supplying 12V DC to the Peltier and fan.
3. **Peltier and Fan:**
    - Both connected in parallel to 12V DC output of the adapter.
    - Fan helps dissipate heat from the Peltier's hot side.
4. **Temperature Sensing & Display:**
    - LM35 sensor analog output connected to ADC0 of ATmega32.
    - Temperature readings are processed and displayed on a 16×2 LCD via PORTD/PORTB lines.
5. **Thermal Management:**
    - Peltier cold side cools the beverage.
    - Hot side attached to a heatsink + fan assembly with thermal paste to improve conduction.

Software Overview

The ATmega32 microcontroller runs the control program, which:

1. Reads temperature from LM35 through the ADC0 channel.
2. Compares the measured temperature with a predefined threshold (e.g., 8°C).
3. Activates PB0 to switch the relay ON if the temperature exceeds the threshold.
4. Deactivates the relay when the temperature drops below the threshold, turning the Peltier and fan OFF.
5. Displays real-time temperature on a 16×2 LCD.

Key Notes

- **ADC Resolution:** 10-bit (0–1023), converted to °C using LM35 scaling (10 mV/°C).

- **Relay Control:** PB0 logic HIGH energizes 5V relay → switches AC → powers Peltier & fan.
- **LCD Display:** 16×2 LCD shows real-time temperature and status (ON/OFF).
- **Safety:** AC switching via relay keeps microcontroller isolated from high voltage.

Code Explanation

The program, written in AVR assembly, handles sensor reading, relay control, and LCD updates.

Initialization

- **PB0** → Output for relay control.
- **ADC** → Configured for LM35 input.
- **LCD** → Initialized (16×2) to display temperature and status.
- Stack pointers and ports are configured for operation.

Temperature Reading & Conversion

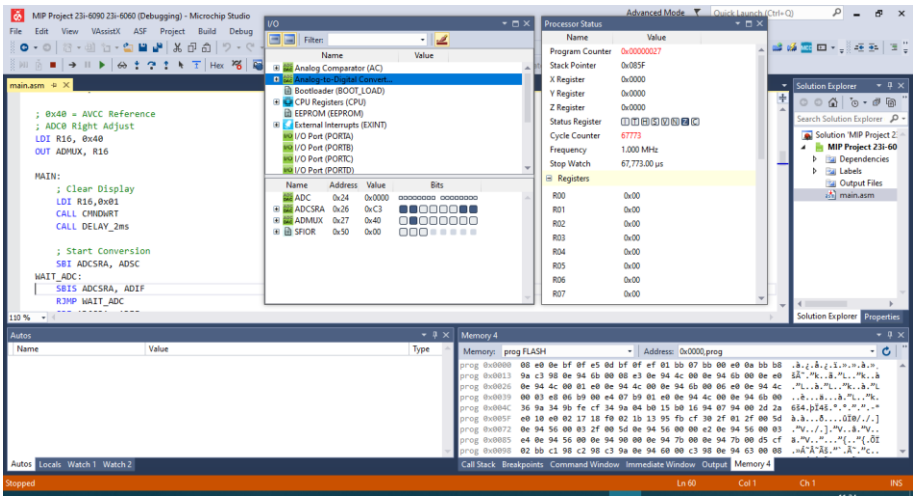
- ADC reads LM35 voltage.
- Temperature calculation: Temp (°C) = ADC_Value / 2 (simplified in code).
- Temperature converted to ASCII for display.

Decision Logic (Relay Control)

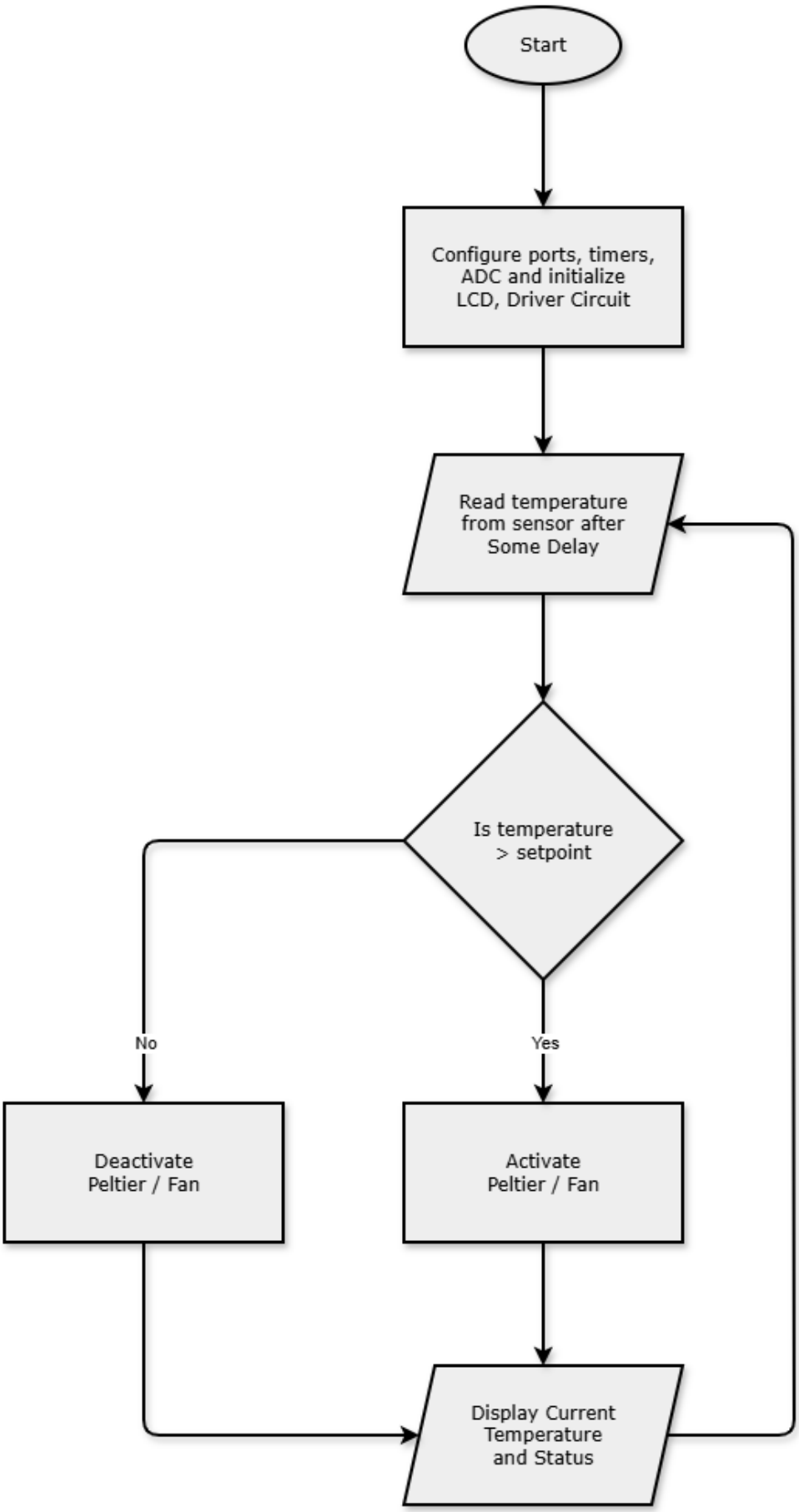
- If Temperature >= Threshold (9 in code ≈ 8°C) → PB0 HIGH → relay ON → Peltier + fan ON.
- Else → PB0 LOW → relay OFF → Peltier + fan OFF.

Display Update

- Tens and ones of temperature displayed on LCD.
- Status “Cooling ON/OFF” can be shown based on relay state.



Flow Chart


Start
Configure ports, timers, ADC and initialize LCD, Driver Circuit
Read temperature from sensor after Some Delay
Is temperature > setpoint
No
Yes
Deactivate Peltier / Fan
Activate Peltier / Fan
Display Current Temperature and Status

Code:

Part (1)

```
.INCLUDE "M32DEF.INC"

; LCD NAMING INIT
.EQU LCD_DPRT = PORTD
.EQU LCD_DDDR = DDRD
.EQU LCD_CPRT = PORTB
.EQU LCD_CDDR = DDRB
.EQU LCD_RS = 1
.EQU LCD_RW = 2
.EQU LCD_EN = 3

; STACK INIT
LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16

; PORT INIT
LDI R16,0xFF
OUT LCD_DDDR, R16
OUT LCD_CDDR, R16
LDI R16,0x00
OUT DDRA, R16
SBI DDRB, 0


CBI LCD_CPRT, LCD_EN
CALL DELAY_2ms

; LCD INIT
LDI R16,0x38
CALL CMNDWRT
CALL DELAY_2ms
LDI R16,0x0E
CALL CMNDWRT
LDI R16,0x01
CALL CMNDWRT
CALL DELAY_2ms
LDI R16,0x06
CALL CMNDWRT

; ADC INIT
LDI R16, 0x83
OUT ADCSRA, R16

; 0x40 = AVCC Reference
; ADC0 Right Adjust
LDI R16, 0x40
OUT ADMUX, R16

MAIN:
    ; Clear Display
    LDI R16,0x01
    CALL CMNDWRT
    CALL DELAY_2ms

    ; Start Conversion
    SBI ADCSRA, ADSC
WAIT_ADC:
```

Part (2)

```
    SBIS ADCSRA, ADIF
    RJMP WAIT_ADC
    SBI ADCSRA, ADIF

    ; Read ADC
    IN R0, ADCL
    IN R1, ADCH

    ; Temp Calculation (ADC / 2)
    LSR R1
    ROR R0
    ; R0 now has Temperature

    ; Convert to ASCII
    MOV R16, R0
    LDI R18, 10
    LDI R17, 0 ; Tens Count

GET_TENS:
    CP R16, R18
    BRCS DONE_TENS
    SUB R16, R18
    INC R17
    RJMP GET_TENS

DONE_TENS:
    ; Display Tens
    MOV R19, R16 ; Save Ones
    MOV R16, R17
    SUBI R16, -48 ; Convert to ASCII
    CALL DATAWRT

    ; Display Ones
    MOV R16, R19
    SUBI R16, -48 ; Convert to ASCII
    CALL DATAWRT

    ; Display ' C'
    LDI R16, ' '
    CALL DATAWRT
    LDI R16, 'C'
    CALL DATAWRT

       CALL CHECK_TEMP_LIMIT

       ; CALL DELAY_5m
       CALL DELAY_1s
       CALL DELAY_1s

RJMP MAIN


CMNDWRT:
    OUT LCD_DPRT, R16
    CBI LCD_CPRT, LCD_RS
    CBI LCD_CPRT, LCD_RW
    SBI LCD_CPRT, LCD_EN
    CALL SDELAY
    CBI LCD_CPRT, LCD_EN
```

Part (3)

```
    CALL DELAY_100_us
RET

DATAWRT:

    OUT LCD_DPRT, R16
    SBI LCD_CPRT, LCD_RS
    CBI LCD_CPRT, LCD_RW
    SBI LCD_CPRT, LCD_EN
    CALL SDELAY
    CBI LCD_CPRT, LCD_EN
    CALL DELAY_100_us
RET


SDELAY:
       NOP
       NOP
RET

DELAY_100_us:
       PUSH R17
       LDI R17,60
DR0:
       CALL SDELAY
       DEC R17
BRNE DR0
       POP R17
RET

DELAY_2ms:
       PUSH R17
       LDI R17,20
LDR0:
       CALL DELAY_100_us
       DEC R17
BRNE LDR0
       POP R17
RET

DELAY_4ms:
       PUSH R17
       LDI R17,2
LDR1:
       CALL DELAY_2ms
       DEC R17
BRNE LDR1
       POP R17
RET

DELAY_1s:
       PUSH R17
       LDI R17,250
LDR2:
       CALL DELAY_4ms
       DEC R17
BRNE LDR2
       POP R17
RET

DELAY_2m:
```

Part (4)

```
    PUSH R20
    PUSH R21
    LDI R20, 2
LOOP_MINS:
    LDI R21, 60
LOOP_SECS:
    CALL DELAY_1s
    DEC R21
    BRNE LOOP_SECS
    DEC R20
    BRNE LOOP_MINS
    POP R21
    POP R20
RET


CHECK_TEMP_LIMIT:
    PUSH R20
    MOV R20, R0
    CPI R20, 15
    BRSH TURN_ON
    CBI PORTB, 0
    RJMP END_CHECK
TURN_ON:
    SBI PORTB, 0
END_CHECK:
    POP R20
RET
```

Hardware pics:

Results

1) Relay reliably switched the Peltier and fan ON/OFF based on temperature.
2) Temperature sensor accurately measured the beverage temperature.
3) LCD displayed real-time temperature and system status.
4) Cooling effect was observable; Peltier successfully lowered beverage temperature while fan dissipated heat from the hot side.

Analysis

1) **System Performance:** The closed-loop system-maintained temperature within set limits efficiently.
2) **Energy Efficiency:** Only activated cooling when needed; PWM control (if implemented) further reduces power usage.
3) **Safety:** Relay ensures microcontroller is isolated from AC supply.
4) **Practical Challenges:** Proper thermal management, sensor calibration, and microcontroller timing were crucial.
5) **Innovation:** Portable, low-cost thermoelectric cooling instead of conventional refrigeration; demonstrates practical embedded system design.