KONGU ENGINEERING COLLEGE

(Autonomous)

Perundurai, Erode – 638 060

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

AUTOMATIC TEMPERATURE - CONTROLLED FAN USING AT89C51 MICROCONTROLLER

A MICRO PROJECT REPORT

For

22ECL42 - Microprocessor and Microcontroller Laboratory

Submitted by

HARIPRASAD M (23ECR068)

KISHOREKUMAR S (23ECR117)

KRISHNAKUMAR VR (23ECR118)

ABSTRACT :

This project is about building an **automatic fan speed controller** using the **8051 microcontroller**. The main idea is to automatically change the speed of a fan depending on the temperature in the room. This helps save energy and removes the need to manually adjust the fan speed every time the temperature changes.

To do this, we use a temperature sensor called **LM35**, which constantly checks the temperature. The sensor sends an analog signal based on how hot or cold the room is. This signal is then converted into a digital format using an **ADC0804** (Analog to Digital Converter), so the microcontroller can understand it. Once the microcontroller gets the temperature data, it decides how fast the fan should run.

The system works in three temperature ranges :

1. **Below 25°C** – The fan is turned **off** or kept at **low speed**, because the room is already cool.
2. **Between 25°C and 30°C** – The fan runs at a **medium speed** to provide a light breeze.
3. **Above 30°C** – The fan switches to **high speed** to help cool the room more effectively.

We use a technique called **PWM (Pulse Width Modulation)** to control the fan speed. This means we turn the fan on and off very quickly, adjusting how long it stays on or off in each cycle. This allows us to control the speed smoothly without using complicated hardware.

We also connected an **LCD display** to the microcontroller to show the current temperature and the fan's status. For example, it might display “TEMP: 28°C” on the top line and “FAN: MED SPEED” on the bottom line. This helps users see what the system is doing in real time.

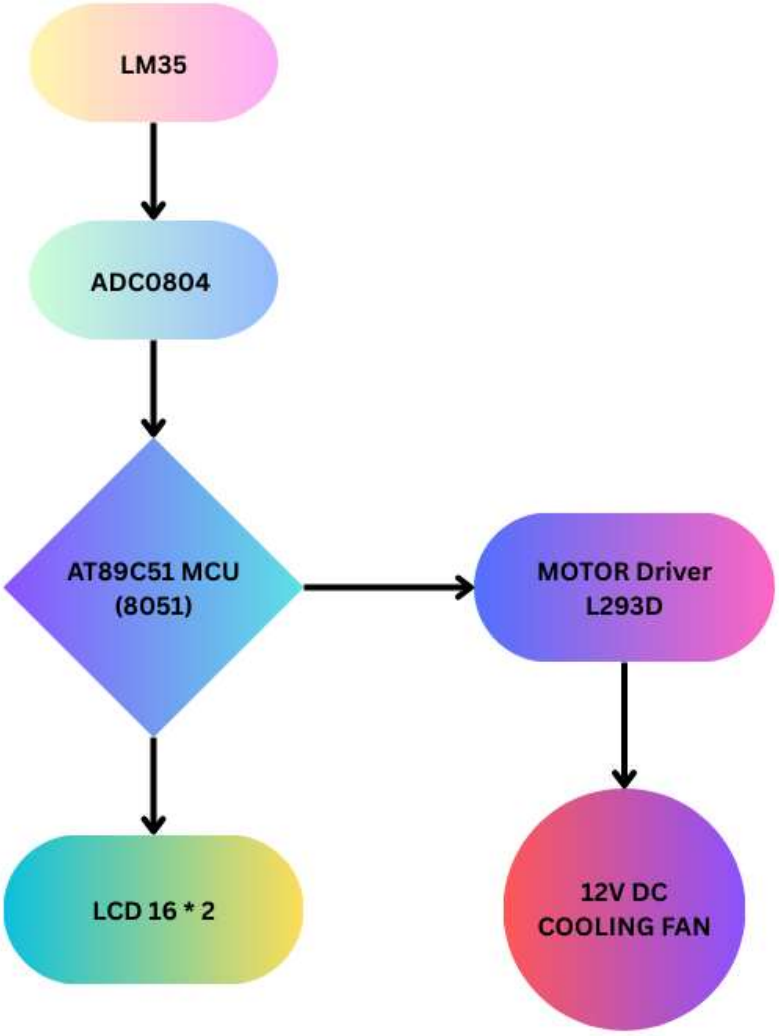
What makes this project special is that it doesn’t need any internet connection or mobile app. It works completely on its own using only the microcontroller and the connected components. It’s a simple, cost-effective solution for keeping room temperature comfortable, especially in places where automatic climate control systems aren’t available.

In short, this project shows how microcontrollers, sensors, and basic electronics can work together to solve a real-world problem. It’s a great example of how we can use technology to make daily life easier and more efficient.

INTRODUCTION :

This project involves the design and implementation of an automatic fan control system using the **AT89C51 microcontroller**. The system automatically controls the fan speed based on the ambient temperature measured via the **LM35 temperature sensor**, digitized using **ADC0804**, and displayed on an **LCD**. The fan operates in three modes: OFF, MEDIUM (PWM control), and HIGH based on the sensed temperature.

BLOCK DIAGRAM :



COMPONENTS :

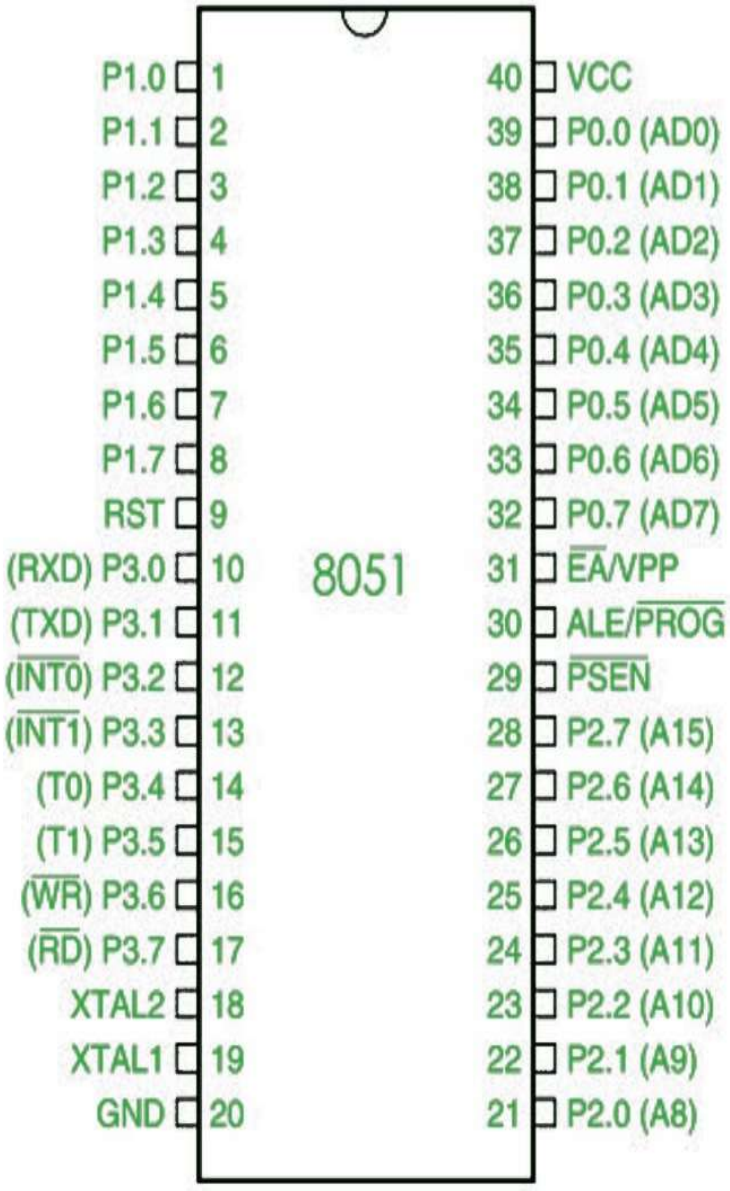
- AT89C51 Microcontroller
- LM35 Temperature Sensor
- ADC0804 (Analog-to-Digital Converter)
- 16x2 LCD Display
- L293D Motor Driver

- 12V DC Cooling Fan
- Power Supply (5V/12V)
- Resistors, Capacitors, and Crystal Oscillator
- Proteus 8.17 Software (for simulation)

Component Explanation :

1) **AT89C51 Microcontroller :** The **AT89C51** is an 8-bit microcontroller from the 8051 family by Atmel. It features 4KB Flash, 128 bytes RAM, 32 I/O lines, two timers, serial communication support, and six interrupt sources. It operates at 5V and uses an external crystal (usually 11.0592 MHz or 12 MHz) for timing. It's highly popular in embedded systems for controlling devices like sensors, motors, and displays due to its simple architecture and wide peripheral support.

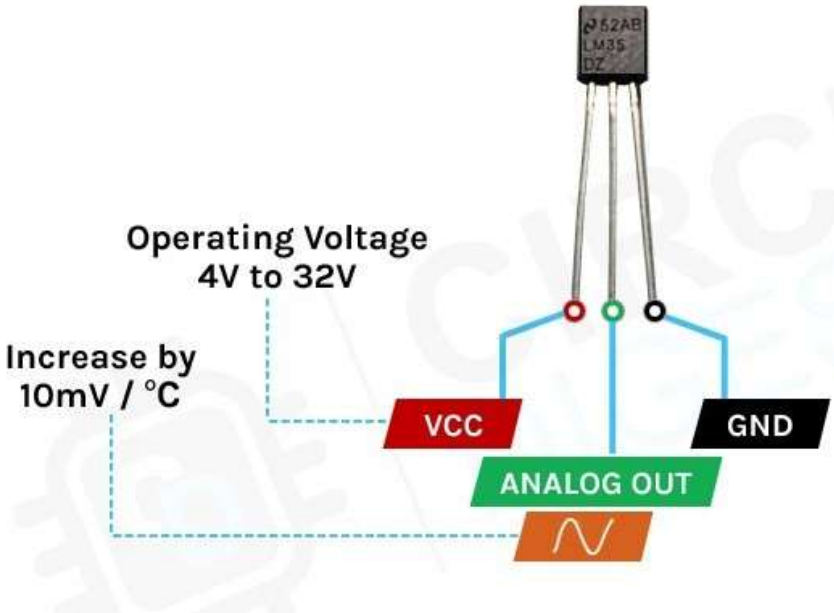
   This microcontroller executes instructions based on input signals and performs tasks like reading sensor data, converting it, and sending control signals to devices like fans or displays.



Pinout Description :

- **Ports :**
    a. **Port 0 (Pins 32–39) :** Multiplexed Address/Data bus (used for ADC data input).
    b. **Port 1 (Pins 1–8) :** General-purpose I/O (used for LCD control).
    c. **Port 2 (Pins 21–28) :** Higher address lines (can be general I/O too).
    d. **Port 3 (Pins 10–17) :** Special purpose (RD, WR, INT, TXD/RXD).
- **Pin 9 :** Reset input
- **Pin 18 & 19 :** XTAL1, XTAL2 for connecting the external crystal oscillator
- **Pin 20 :** GND
- **Pin 40 :** Vcc (5V power supply)

2) **LM35 Temperature Sensor :** The **LM35** is a precision analog temperature sensor with output voltage linearly proportional to Celsius temperature. It provides 10mV/°C, meaning at 25°C it gives 250mV. It has low self-heating and works from 4V to 30V. It doesn't require any calibration or external components for standard use.
It is ideal for this project because its analog output can be accurately converted to digital by the ADC0804 and processed by the AT89C51 to determine fan control actions.



Pinout Description :

- **Pin 1 (Vcc)** : +5V supply
- **Pin 2 (Vout)** : Analog output → Connect to **Vin+ (Pin 6)** of ADC0804
- **Pin 3 (GND)** : Ground

3) **ADC0804 – Analog to Digital Converter :** The **ADC0804** converts analog signals (like LM35 output) into an 8-bit digital value readable by microcontrollers. It operates with a 5V supply and

takes differential analog input through Vin+ and Vin−. It has control signals like RD, WR, and INTR for managing data conversion and interfacing with microcontrollers.

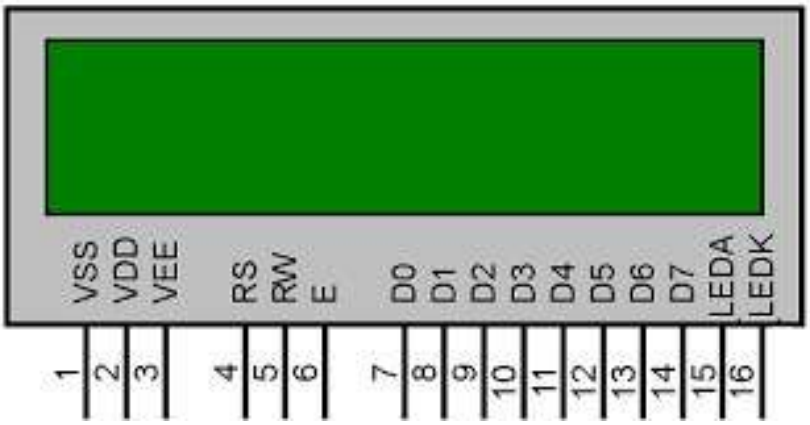
This component is essential as the AT89C51 does not have a built-in ADC. ADC0804 bridges this gap by converting the temperature signal from LM35 into digital form.

Pinout Description :

- **Pin 1 (CS)** : Chip Select (connect to GND to always enable)
- **Pin 2 (RD)** : Read control → Connect to P3.7
- **Pin 3 (WR)** : Write control → Connect to P3.6
- **Pin 4 (CLK IN)** : External clock input (connect to RC circuit)
- **Pin 5 (INTR)** : Interrupt output → Connect to P3.2 (indicates conversion complete)
- **Pin 6 (Vin+)** : Connect to LM35 output
- **Pin 7 (Vin−)** : Ground
- **Pins 11–18 (DB7–DB0)** : Digital output → Connect to Port 0 (P0.0–P0.7)
- **Pin 9 (Vref/2)** : For reference voltage (leave open for default 5V)
- **Pins 10 & 8** : Ground
- **Pin 20** : Vcc (5V supply)

4) **16x2 LCD Display (HD44780) :** The **16x2 LCD** is used to display temperature values, fan status, or any messages. It has 16 pins and can show 2 lines of 16 characters each. It supports 8-bit and 4-bit communication with microcontrollers, and is controlled using RS, RW, and EN pins along with data lines.

   In this project, the LCD is used to display the current temperature (in °C) and fan speed status (e.g., OFF, MEDIUM, HIGH).

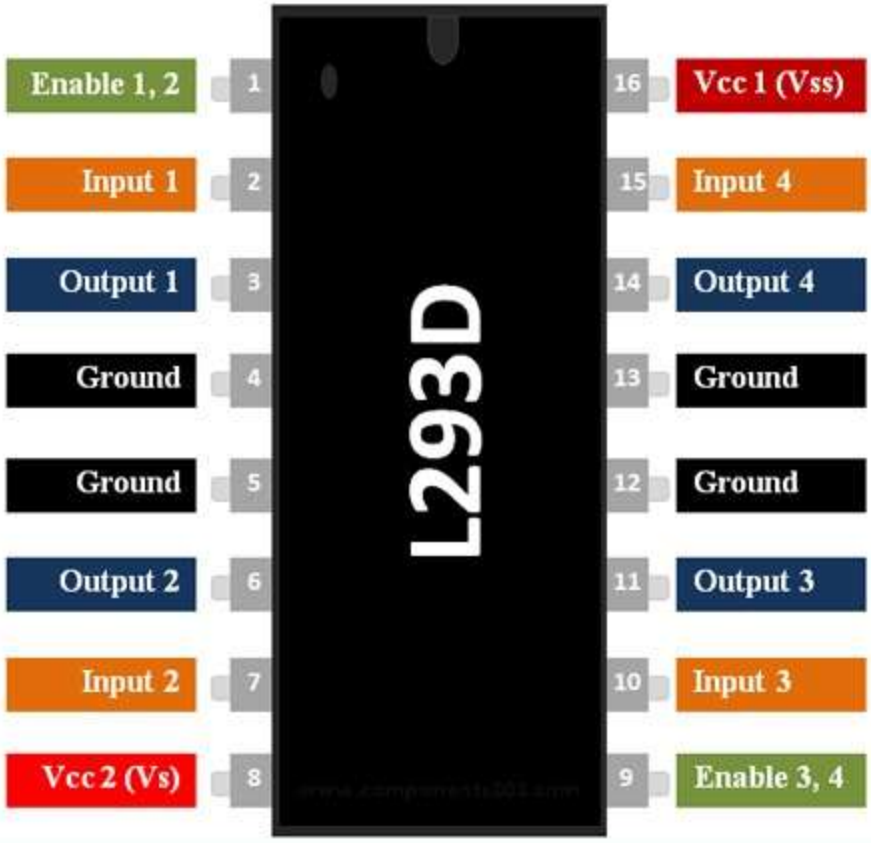


Pinout Description :

- **Pin 1 (VSS)** : GND
- **Pin 2 (VCC)** : +5V
- **Pin 3 (VEE)** : Contrast control via potentiometer
- **Pin 4 (RS)** : Register select → Connect to P1.x
- **Pin 5 (RW)** : Read/Write → GND (write mode)
- **Pin 6 (EN)** : Enable → Connect to P1.x
- **Pins 7–14 (D0–D7)** : Data lines (usually only D4–D7 used in 4-bit mode)
- **Pin 15 (LED+) & Pin 16 (LED−)** : Backlight (optional)

5) **L293D Motor Driver :** The L293D is a dual H-bridge motor driver IC used to control the direction and speed of DC motors. It can control two DC motors or one stepper motor. It takes logic inputs and supplies higher current output to the motors. It supports PWM for speed control and is perfect for driving a fan using control signals from the microcontroller.

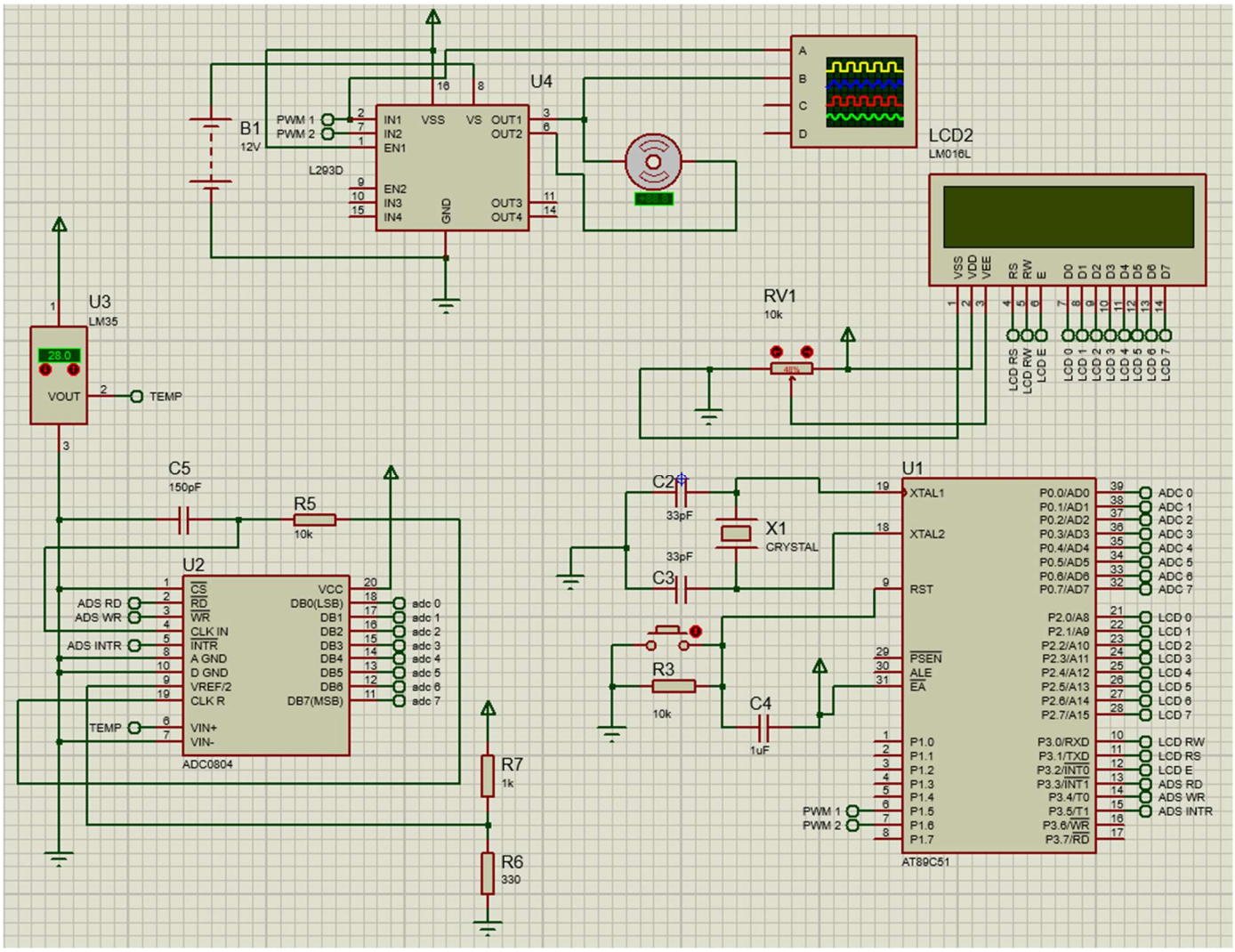
   In this project, the AT89C51 sends logic high/low or PWM signals to L293D to control fan speed based on the temperature.



Pinout Description :

- **Pins 1 & 9 (EN1, EN2)** : Enable pins → Connect to 5V (or use PWM for speed control)
- **Pins 2 & 7 (IN1, IN2)** : Logic inputs for Motor 1 → Connect to P1.5, P1.6
- **Pins 10 & 15 (IN3, IN4)** : Logic inputs for Motor 2 (not used here)
- **Pins 3 & 6 (OUT1, OUT2)** : Motor outputs
- **Pins 4, 5, 12, 13** : GND
- **Pin 8** : Motor supply voltage (Vcc2) → Connect to 9V or 12V (for fan)
- **Pin 16** : Logic supply voltage (Vcc1) → 5V

CIRCUIT DIAGRAM :



CODE :

```
ORG 0000H                ; Start program from address 0000H
LJMP MAIN                ; Jump to the MAIN label (start of main code)

ORG 000BH                ; Timer 0 interrupt vector address
LJMP TIMER_ISR           ; Jump to TIMER_ISR when timer interrupt occurs

; ----------- RAM Variable Definitions ------------
PWM_FLAG    EQU 32H      ; Flag to toggle PWM ON/OFF
TEMP_VAL    EQU 33H      ; Stores ADC temperature value
ON_DELAY    EQU 31H      ; ON time delay for PWM
OFF_DELAY   EQU 30H      ; OFF time delay for PWM

MAIN:
    MOV P0, #0FFH        ; Set P0 as input (for ADC data)
    SETB P1.5            ; Set P1.5 high (fan control pin)
    CLR P1.6             ; Set P1.6 low (other fan pin)
```

```
17
18      ; LCD Initialization
19      MOV A,#38H              ; LCD 2 lines, 5x7 matrix
20      ACALL CMD               ; Send command to LCD
21
22      ACALL DELAY             ; Wait for LCD to process
23      MOV A,#0EH              ; LCD ON, cursor ON
24      ACALL CMD               ; Send command
25
26      ACALL DELAY             ; Wait again
27      MOV A,#01H              ; Clear LCD screen
28      ACALL CMD               ; Send command
29      ACALL DELAY             ; Wait after clearing
30
31      MOV A,#080H             ; Move LCD cursor to 1st row, 1st column
32      ACALL CMD               ; Send command
33
34      ; Welcome message
35      MOV DPTR, #MSG1         ; Point to first message
36      ACALL WRITESTR          ; Display message
37      MOV A,#0C0H             ; Move to 2nd row
38      ACALL CMD               ; Send command
39      MOV DPTR, #MSG2         ; Point to second message
40      ACALL WRITESTR          ; Display message
41      ACALL DELAY1            ; Wait to keep message visible
42      MOV A,#01H              ; Clear LCD
43      ACALL CMD               ; Send command
44
45      ; Display Labels
46      MOV A,#080H             ; First line start
47      ACALL CMD               ; Send command
48      MOV DPTR, #TEMPMSG      ; Point to "TEMP: " string
49      ACALL WRITESTR          ; Display temperature label
50      MOV A,#0C0H             ; Second line start
51      ACALL CMD               ; Send command
52      MOV DPTR, #FANMSG       ; Point to "FAN : " string
53      ACALL WRITESTR          ; Display fan label
54
55  MAIN_LOOP:
56      ; Start ADC0804 Conversion
57      SETB P3.5               ; Dummy INTR high (no effect, possibly error)
58      SETB P3.3               ; WR = 1 (standby)
59      CLR P3.4                ; RD = 0 (start read)
60      SETB P3.4               ; RD = 1 (end read)
61
62  WAIT_ADC:
63      JB P3.5, WAIT_ADC       ; Wait until INTR pin is low
64      CLR P3.3                ; WR = 0 (start conversion)
65      MOV TEMP_VAL, P0        ; Read ADC data into TEMP_VAL
```

```
66
67      ; Display Temperature
68      MOV A,#086H             ; LCD cursor to line 1, column 6
69      ACALL CMD               ; Send command
70      ACALL SHOW_TEMP         ; Show temperature on LCD
71
72      ; Fan control logic
73      MOV A, TEMP_VAL         ; Move ADC value to A
74      CJNE A, #25, CHK25      ; Compare with 25°C
75      LJMP FAN_OFF            ; If exactly 25, turn fan off
76
77  CHK25: JC FAN_OFF           ; If less than 25, turn off
78      CJNE A, #30, CHK30      ; Compare with 30°C
79      LJMP FAN_ON             ; If exactly 30, fan ON
80
81  CHK30: JC FAN_PWM           ; If between 25-30, use PWM
82      LJMP FAN_ON             ; If >30, turn fan ON
83
84  FAN_OFF:
85      CLR TR0                 ; Stop Timer0
86      MOV IE, #80H            ; Disable all interrupts except external
87      CLR P1.5                ; Turn off fan (logic 0)
88      CLR P1.6                ; Clear other fan pin
89      MOV A,#0C6H             ; Cursor to 2nd row, col 6
90      ACALL CMD               ; Send command
91      MOV DPTR, #LOWMSG       ; "LOW SPEED"
92      ACALL WRITESTR          ; Show on LCD
93      LJMP CONTINUE           ; Go back to main loop
94
95  FAN_PWM:
96      MOV R0, #ON_DELAY       ; Point to ON delay location
97      MOV A, #80H             ; Load ON time value
98      MOV @R0, A              ; Store in ON_DELAY
99      MOV R0, #OFF_DELAY      ; Point to OFF delay
100     MOV A, #80H             ; Load OFF time
101     MOV @R0, A              ; Store in OFF_DELAY
102     ACALL ENABLE_PWM        ; Start PWM
103     MOV A,#0C6H             ; LCD 2nd row, col 6
104     ACALL CMD               ; Send command
105     MOV DPTR, #MEDMSG       ; "MED SPEED"
106     ACALL WRITESTR          ; Show on LCD
107     LJMP CONTINUE           ; Go back to main loop
108
109 FAN_ON:
110     CLR TR0                 ; Stop Timer0
111     MOV IE, #80H            ; Disable PWM
112     SETB P1.5               ; Turn ON fan
113     CLR P1.6                ; Other fan pin
114     MOV A,#0C6H             ; LCD row 2 col 6
```

```
    ACALL CMD              ; Send command
    MOV DPTR, #HIGHMSG     ; "HIG SPEED"
    ACALL WRITESTR         ; Show on LCD
    LJMP CONTINUE          ; Go back to main loop

ENABLE_PWM:
    MOV TMOD, #01H         ; Set Timer0 in Mode 1 (16-bit)
    MOV R0, #ON_DELAY      ; Load ON delay
    MOV A, @R0
    MOV TH0, A             ; Set high byte
    MOV TL0, #00H          ; Set low byte
    MOV PWM_FLAG, #01H     ; Start with ON state
    SETB TR0               ; Start Timer0
    SETB P1.5              ; Fan ON
    CLR P1.6               ; Fan direction

    MOV IE, #82H           ; Enable Timer0 interrupt
    RET                    ; Return from subroutine

CONTINUE:
    ACALL DELAY1           ; Delay before looping again
    LJMP MAIN_LOOP         ; Repeat the process

TIMER_ISR:
    CLR TR0                ; Stop Timer0
    MOV A, PWM_FLAG        ; Check PWM state
    CJNE A, #01H, SET_ON   ; If OFF, set ON
    MOV R0, #OFF_DELAY     ; Load OFF delay
    MOV A, @R0
    MOV TH0, A             ; Set delay
    MOV PWM_FLAG, #00H     ; Next: ON time
    CLR P1.5               ; Turn fan OFF
    SJMP DONE              ; Skip to end

SET_ON:
    MOV R0, #ON_DELAY      ; Load ON delay
    MOV A, @R0
    MOV TH0, A             ; Set delay
    MOV PWM_FLAG, #01H     ; Next: OFF time
    SETB P1.5              ; Turn fan ON

DONE:
    MOV TL0, #00H          ; Reset low byte
    SETB TR0               ; Restart timer
    RETI                   ; Return from interrupt

SHOW_TEMP:
    MOV A, TEMP_VAL        ; Load temperature value
```

```
    MOV B, #10          ; Divide by 10
    DIV AB              ; A = tens, B = units
    MOV R7, B           ; Save units
    ADD A, #30H         ; Convert tens to ASCII
    ACALL WRITE         ; Display tens digit
    MOV A, R7           ; Load units
    ADD A, #30H         ; Convert to ASCII
    ACALL WRITE         ; Display units
    MOV A, #0DFH        ; Display degree symbol
    ACALL WRITE
    MOV A, #'C'         ; Display 'C'
    ACALL WRITE
    RET

CMD:
    MOV P2,A            ; Send command byte to LCD
    CLR P3.1            ; RS = 0 (command mode)
    CLR P3.0            ; RW = 0 (write mode)
    SETB P3.2           ; Enable high
    ACALL DELAY         ; Wait
    CLR P3.2            ; Enable low (latch command)
    RET

WRITE:
    MOV P2,A            ; Send data byte to LCD
    SETB P3.1           ; RS = 1 (data mode)
    CLR P3.0            ; RW = 0 (write)
    SETB P3.2           ; Enable high
    ACALL DELAY         ; Wait
    CLR P3.2            ; Enable low (latch data)
    RET

WRITESTR:
    CLR A               ; Clear A for index
    MOVC A,@A+DPTR      ; Get character from code memory
    JZ RETURN           ; If 0, end of string
    ACALL WRITE         ; Display character
    ACALL DELAY         ; Wait
    INC DPTR            ; Move to next char
    SJMP WRITESTR       ; Repeat
RETURN: RET             ; Return from subroutine

DELAY:
    MOV TMOD,#01H       ; Timer0 mode 1
    MOV TH0,#0EDH       ; Set delay high byte
    MOV TL0,#097H       ; Set delay low byte
    SETB TR0            ; Start timer
```

```
210 WAIT: JNB TF0, WAIT        ; Wait for timer overflow
211     CLR TR0                ; Stop timer
212     CLR TF0                ; Clear overflow flag
213     RET
214
215 DELAY1:
216     MOV R5, #100           ; Loop 100 times
217
218 DL1: ACALL DELAY            ; Call small delay
219     DJNZ R5, DL1           ; Repeat until R5 = 0
220     RET
221
222 TEMPMSG:  DB "TEMP: ", 0      ; String to show "TEMP: "
223 FANMSG:   DB "FAN : ", 0      ; String to show "FAN : "
224 LOWMSG:   DB "LOW SPEED", 0   ; String for low speed
225 MEDMSG:   DB "MED SPEED", 0   ; String for medium speed
226 HIGHMSG:  DB "HIG SPEED", 0   ; String for high speed
227 MSG1:     DB "AUTO TEMP FAN", 0 ; Startup message line 1
228 MSG2:     DB "8051 (AT89C51)", 0 ; Startup message line 2
229
230 END                          ; End of program
```

Working Principle :

I. System Initialization :

- The program starts at memory address 0000H and jumps to the MAIN label.
- Timer interrupt vector is set at 000BH for PWM handling.
- LCD is initialized to display welcome and status messages.
- Ports are configured:
    - P0 is used to read ADC0804 output (temperature).
    - P1.5 and P1.6 control the fan through an H-bridge (like L293D).
    - P2 and P3 are used for LCD and ADC control signals.

II. Reading Temperature :

- The **ADC0804** is used to read analog voltage from **LM35 temperature sensor**, which outputs 10 mV/°C.
- Control signals:
    - WR and RD lines (P3.3 and P3.4) are used to start and read conversion.
    - INTR (P3.5) is polled to wait for conversion completion.

- The ADC output (8-bit) is stored in TEMP_VAL.

III. Displaying Temperature :

- LCD shows:
    - "TEMP: " on the first line.
    - "FAN : " on the second line.
- The temperature value from TEMP_VAL is converted to ASCII and displayed on the LCD with the degree symbol and 'C'.

IV. Fan Speed Control Logic :

Based on the temperature (TEMP_VAL) :

➤ Below 25°C

- Fan is **OFF**.
- P1.5 and P1.6 are cleared.
- LCD shows: "LOW SPEED".

➤ 25°C to 30°C

- **PWM is enabled**.
- Timer0 toggles P1.5 ON and OFF to control fan speed.
- LCD shows: "MED SPEED".

➤ Above 30°C

- Fan is **fully ON**.
- P1.5 is set; P1.6 is cleared.
- LCD shows: "HIG SPEED".

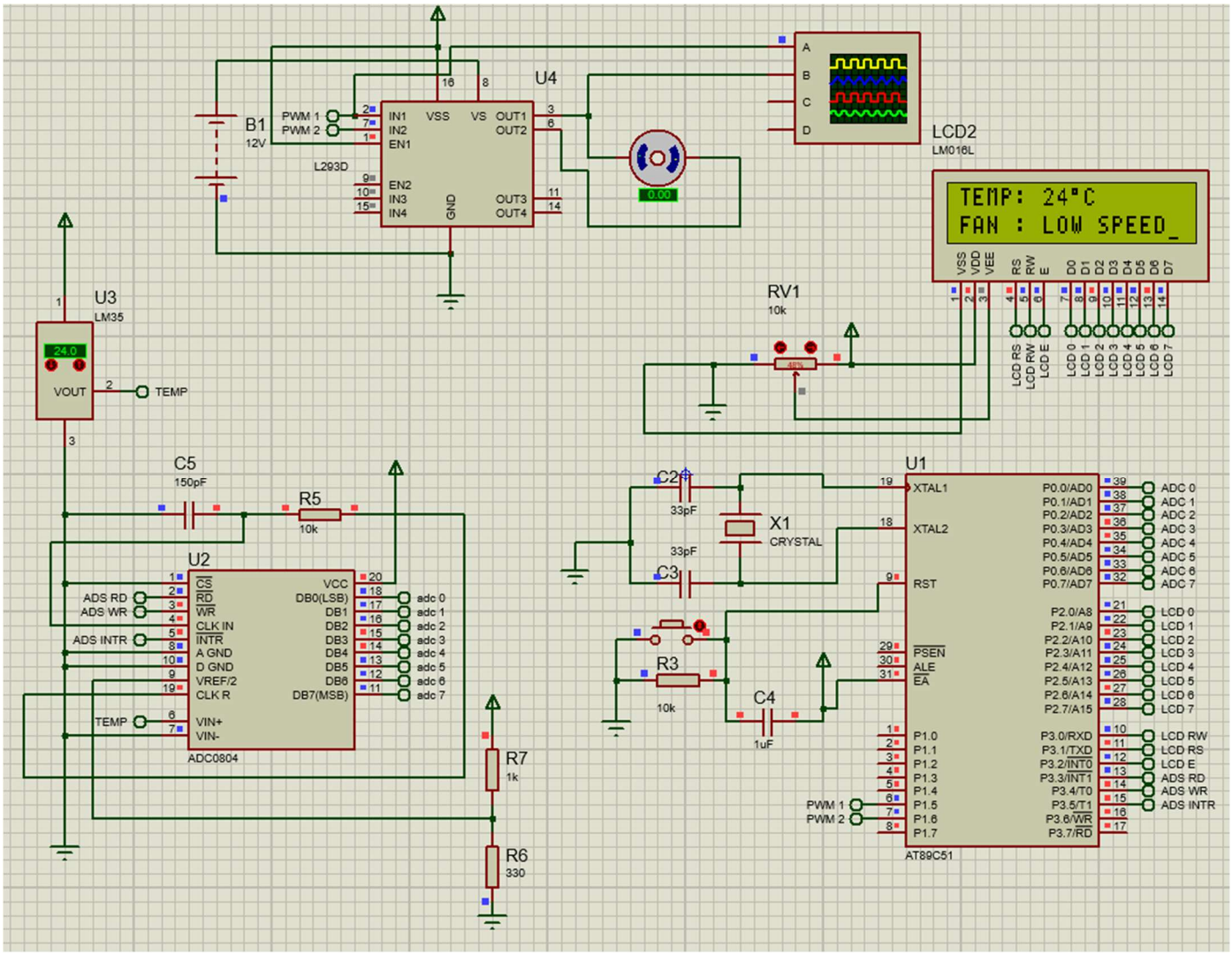
V. PWM Generation Using Timer Interrupt

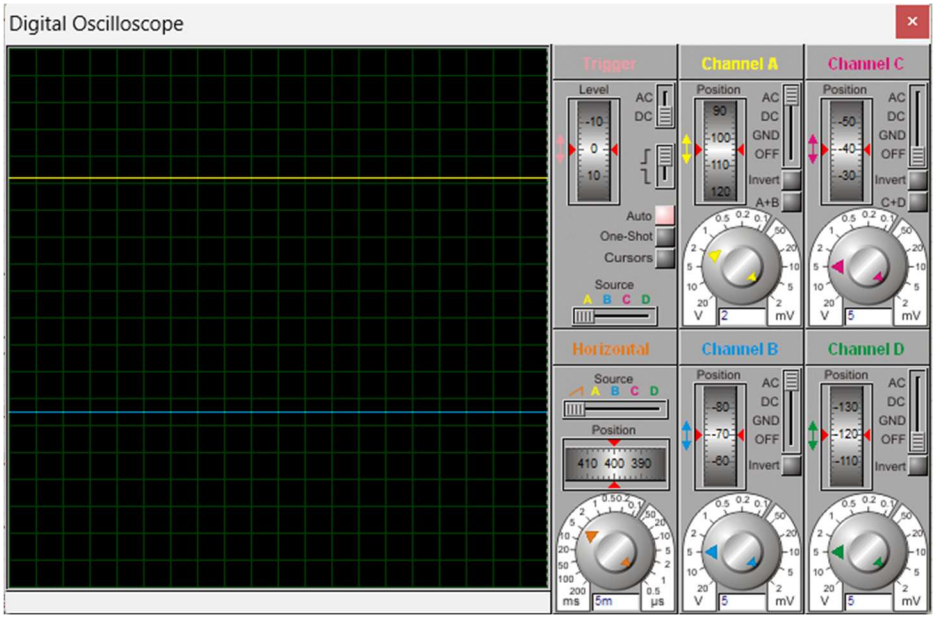
- Timer0 is configured in Mode 1 (16-bit timer).
- Two delays: ON_DELAY and OFF_DELAY control PWM duty cycle.
- TIMER_ISR toggles the fan pin (P1.5) based on PWM_FLAG.
- This creates a square wave signal for medium speed fan operation.

SIMULATION OUTPUT :

I.Low Temperature Condition (< 25°C)

- **Description :**
  - The LM35 sensor provides an analog signal corresponding to a temperature below 25°C (e.g., 24°C).
  - The microcontroller reads this through ADC0804.
  - Based on your logic, the fan is turned OFF, and no PWM signal is generated.
- **Expected Observation :**
  - On the LCD: TEMP: 24°C and FAN : LOW SPEED
  - On the signal analyzer: Flat line (0V PWM) or no pulse train.
  - On the RPM display: RPM = 0
- **Purpose:** To demonstrate energy saving by not operating the fan when cooling is unnecessary.





SCREENSHOT 1

II. Medium Fan Speed (PWM 50% Duty Cycle) :

This screenshot 2 shows the **PWM signal** generated when the room temperature is between **25°C and 30°C**. In this range, the microcontroller activates the fan at **medium speed** using a **50% duty cycle PWM** signal.

Key Details :

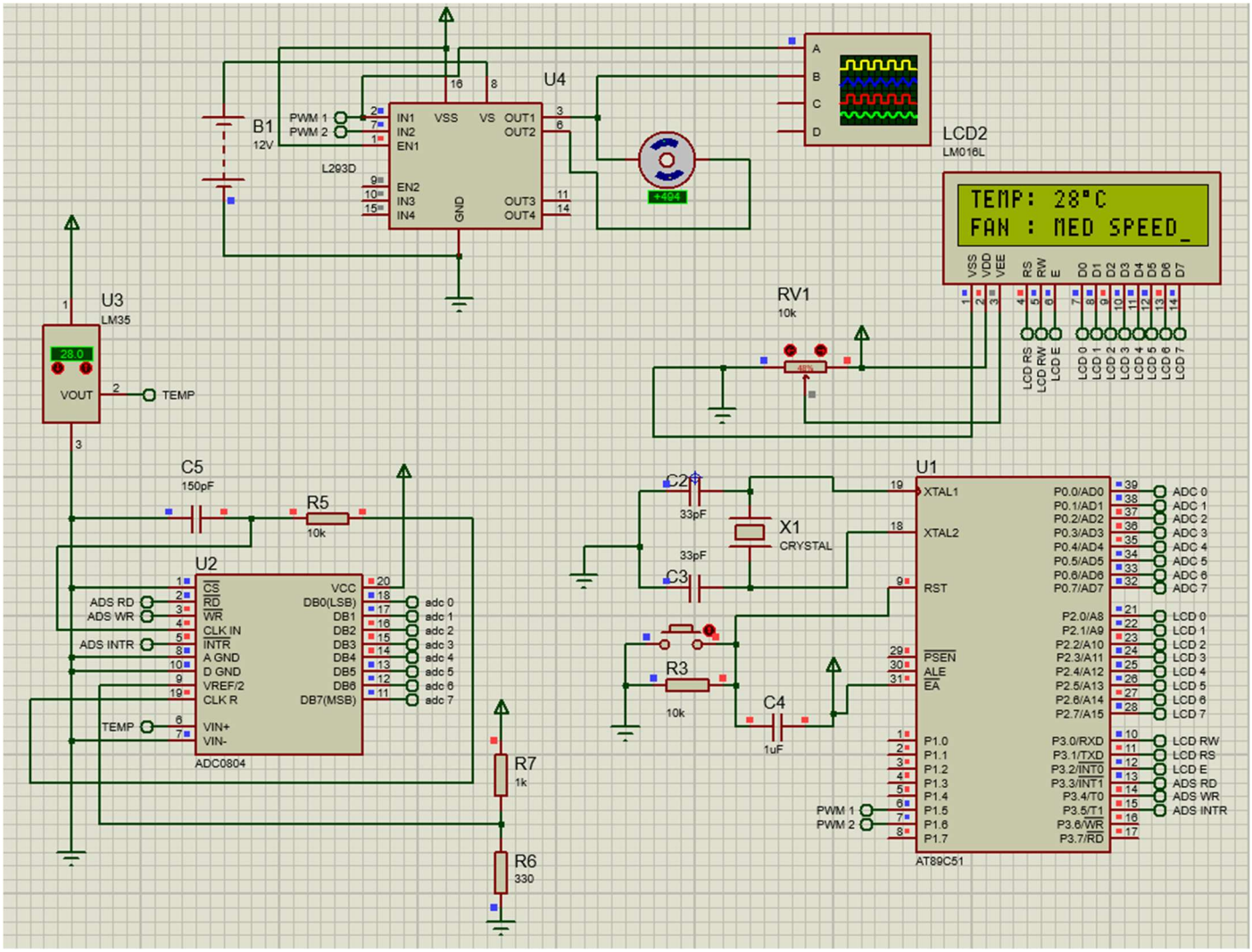
- **Duty Cycle**: 50% Achieved by setting equal ON and OFF time delays in the timer

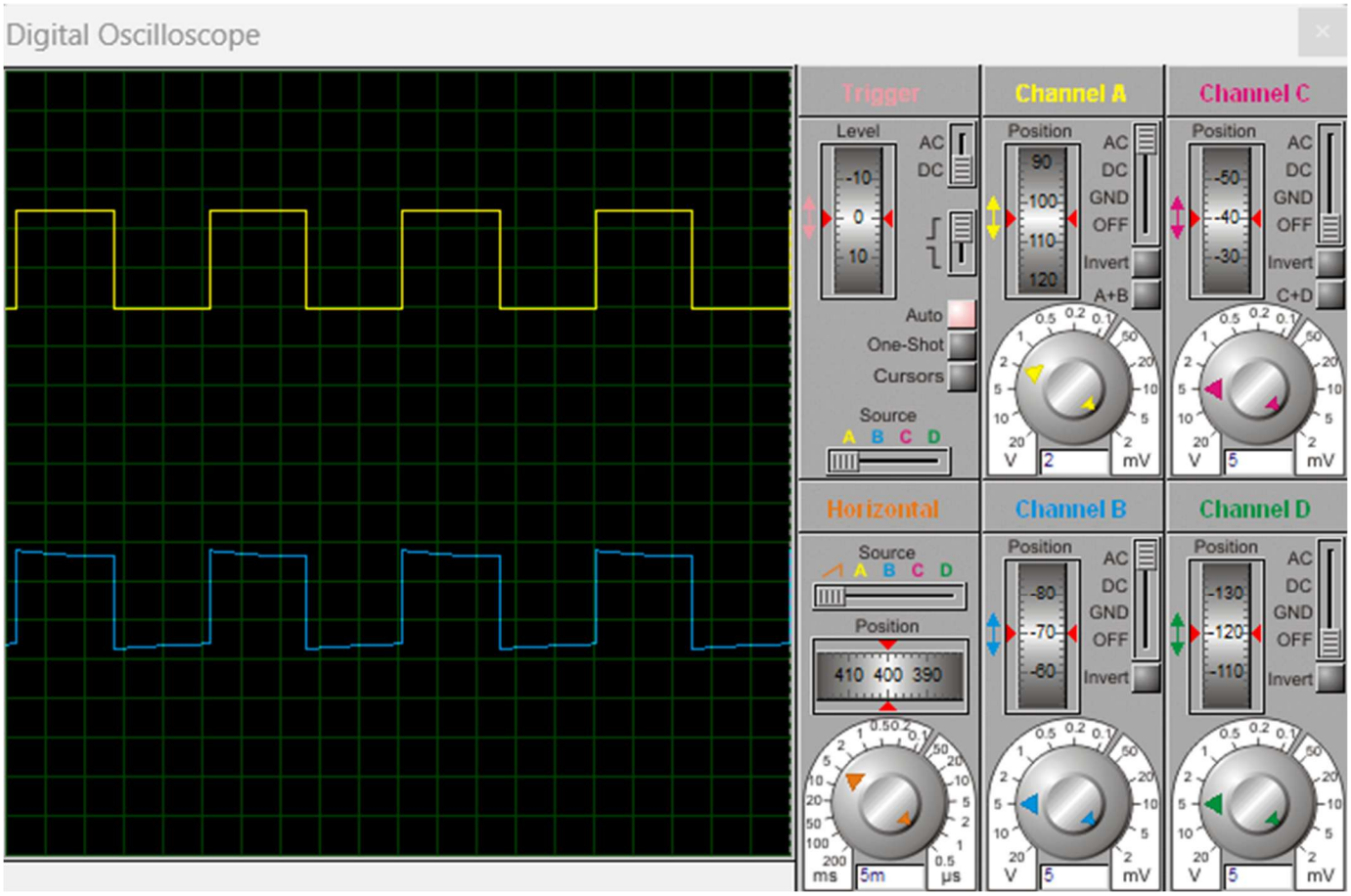
  ON_DELAY = OFF_DELAY = 80H (128 decimal)

- **Calculation** :
  - Timer0 counts from TH0 = 80H (32768) to FFFFH (65535)
  - Delay = 65536 - 32768 = **32768 µs = 32.768 ms**
  - Total PWM Period = 32.768 ms ON + 32.768 ms OFF = **65.536 ms**
  - Duty Cycle = (32.768 / 65.536) × 100 = **50%**

Fan Behavior :

- This 50% duty PWM signal is sent to the fan via the L293D driver.
- The fan receives half the full power, resulting in **medium speed (~494 RPM)**.
- The waveform in the screenshot confirms the equal high and low durations.

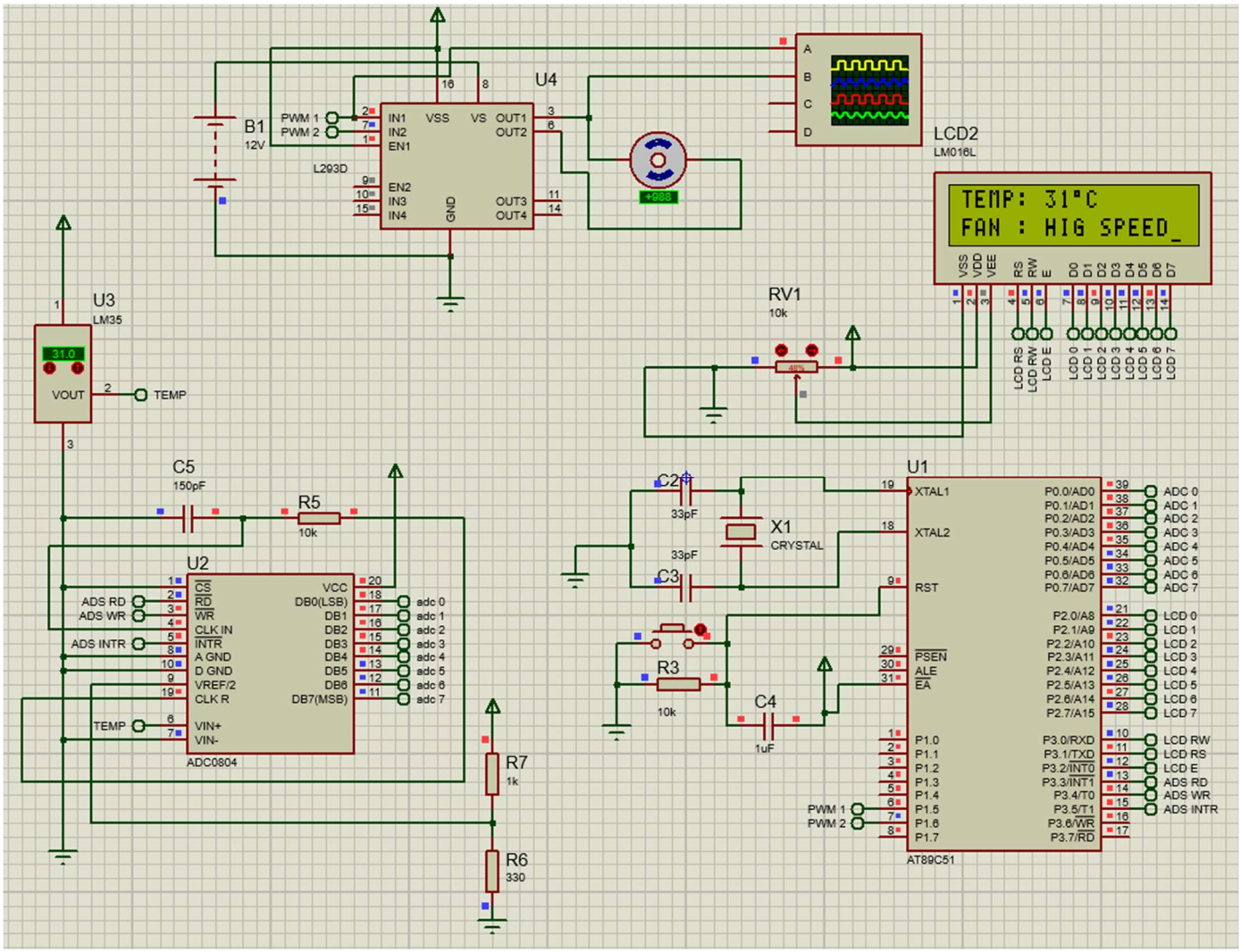


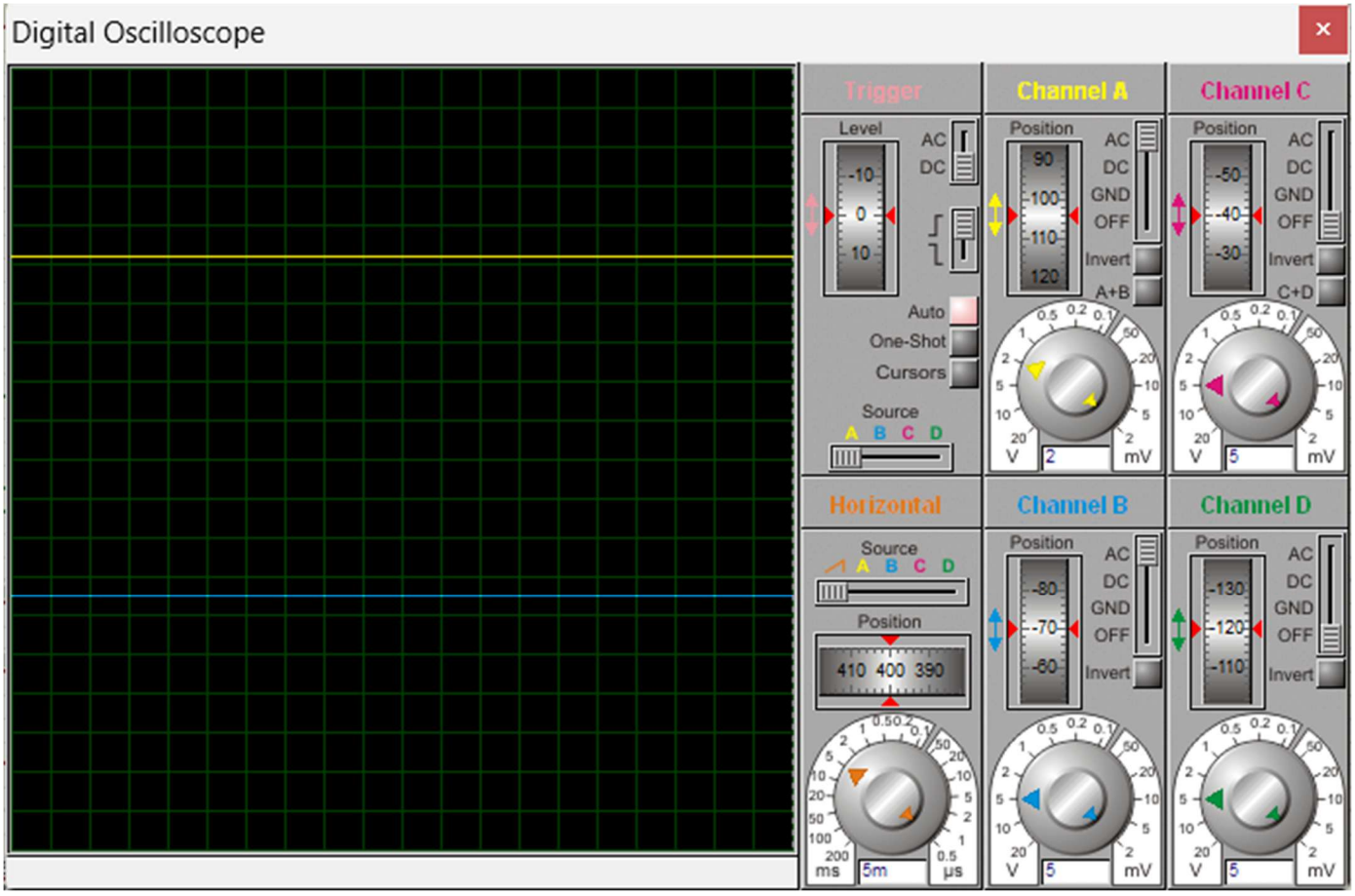


SCREENSHOT 2

III. High Temperature Condition (> 30°C) :

- **Description** :
  - The LM35 outputs voltage for a high temperature, **above 30°C**.
  - The microcontroller drives the fan at full speed (100% duty cycle, i.e., always ON).
- **Expected Observation** :
  - On the LCD: TEMP: 32°C and FAN : HIG SPEED
  - PWM signal: **Constant HIGH signal** (logic high, no modulation).
  - On the RPM display: RPM = 988
- **Purpose**: To ensure the fan runs at max speed to cool the environment effectively.





SCREENSHOT 2

Simulation Observations (from Proteus 8.17) :

Temperature (°C)	Fan Status	PWM Signal	Measured RPM
Below 25	OFF	No PWM / 0V	0
25 – 30	Medium Speed	PWM (50% duty)	494
Above 30	Full Speed	Constant HIGH	988

II.HARDWARE PHOTO :

CONCLUSION :

This project focused on designing and implementing an automatic fan speed controller using the AT89C51 microcontroller. The main aim was to control the speed of a DC fan based on the ambient room temperature. Throughout the development and testing process, I gained valuable practical experience in working with sensors, ADCs, microcontroller interfacing, and pulse width modulation techniques.

The hardware setup included the LM35 temperature sensor, ADC0804 for analog-to-digital conversion, a 16x2 LCD for display, and a DC fan driven by the L293D motor driver IC. The fan speed was controlled according to the temperature input from the LM35 sensor. When the temperature was below 25°C, the fan remained off to save energy. Between 25°C and 30°C, the fan operated at medium speed using a 50% duty cycle. For temperatures above 30°C, the fan ran at full speed to ensure proper cooling. All stages were successfully simulated using Proteus 8.17, and the results matched the expected behavior.

The use of PWM via software (implemented through timer interrupts) was a new and challenging experience. Understanding how to generate square wave signals by adjusting the ON and OFF durations helped me develop a deeper knowledge of digital signal control. Writing the assembly code, debugging it, and seeing it work in the simulation gave me confidence in my microcontroller programming abilities.

This project is particularly useful because it provides a standalone solution that doesn’t rely on external networks, mobile apps, or Wi-Fi. The automation is done entirely through embedded hardware and software, making the system more stable, cost-effective, and easy to maintain. It could be applied in rooms, small data centers, or industrial setups where continuous manual temperature control is not practical.

In conclusion, this project has helped me better understand the concepts of embedded systems and real-time automation. It also taught me how theory and simulation come together to build something practical and functional. It was a highly rewarding experience and one that gave me real insight into how automation systems work in the real world.

OUTCOME :

1. **Project Objective Achieved** :
   The system was successfully designed and implemented to control the speed of a DC fan automatically using the AT89C51 microcontroller based on room temperature.
2. **Temperature Detection Accuracy** :
   The LM35 temperature sensor accurately detected ambient temperatures and provided analog voltage corresponding to the temperature.
3. **ADC0804 Functionality** :
   The ADC0804 converted the analog temperature signal into an 8-bit digital value, which was correctly interpreted by the microcontroller.
4. **Fan Speed Control via PWM** :
   The fan speed was precisely controlled using software-based PWM. Different speeds were achieved by varying the duty cycle:
   - Below 25°C: Fan OFF (0 RPM)
   - Between 25°C–30°C: Medium speed (494 RPM, ~50% duty cycle)
   - Above 30°C: High speed (988 RPM, 100% duty cycle)
5. **LCD Display Functionality** :
   A 16x2 LCD module displayed the current temperature and fan status, making the system user-friendly and informative.
6. **Successful Simulation** :
   All components and logic were tested using Proteus 8.17 simulation software. The system behaved exactly as expected in all three stages.
7. **PWM Signal Verification** :
   The PWM signals for 50% and 100% duty cycles were verified using the oscilloscope tool in Proteus, showing correct signal generation.
8. **Hardware Efficiency** :
   The system operated without the need for Wi-Fi, Bluetooth, or any mobile application. It functioned independently and reliably.
9. **Low Power Consumption**:
   The fan remains OFF when not needed, reducing power consumption, which makes this project energy efficient and eco-friendly.
10. **Skill Development** :
    This project enhanced my understanding of:
    - Microcontroller programming (Assembly)

- Interfacing of ADC, LCD, and sensors
- Timer and interrupt handling
- PWM-based motor control
- Circuit simulation using Proteus

11. **Real-time Application Readiness** :
    The developed system is suitable for practical implementation in homes, offices, or server rooms for automatic cooling.

12. **Overall Satisfaction** :
    The project met all its goals. It gave me valuable practical experience in embedded systems and increased my confidence in handling real-world electronics projects.

KONGU ENGINEERING COLLEGE, PERUNDURAI

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

MINI PROJECT RUBRICS

22ECL42 - Microprocessor and Microcontroller Laboratory

Title : AUTOMATIC TEMPERATURE - CONTROLLED FAN USING AT89C51 MICROCONTROLLER

Roll No	Name of the Student
23ECR068	HARIPRASAD M
23ECR117	KISHOREKUMAR S
23ECR118	KRISHNAKUMAR VR

Content				
Methodology & Innovation (15)				
Simulation/ Implementation & Analysis (15)				
Result (15)				
Report Writing (05)				
Team work, Presentation &Viva (10)				
Total (60)				

Faculty Signature