Abdullah Gül University

Department of Computer Engineering

Embedded Systems Course Final Project Report

Temperature-Controlled Ventilation System for Indoor Environments

Team Members:

Merve Elanur Azman
Gül Nur Deviren
Şeyma Göçmez

Course Instructor: Abdulkadir Gülşen

Contents

1 Project Objective

This project involves a system designed to manage indoor environments by controlling temperature. It works by sensing the current room temperature and showing it on a LCD display. Users can also control an electric motor using simple buttons, allowing them to change its running direction (like forward or reverse) and its speed. This motor is a key part of a fan that helps to adjust the room's temperature and airflow. For the system to work well, save energy, and effectively control the temperature, it's very important that the fan spins in the correct direction. If it spins the wrong way, the system won't be able to manage the environment properly.

1.1 Overview

This project focuses on an indoor control system that reads the ambient temperature via a sensor and displays it on an LCD screen. It also allows the user to control the direction (forward, reverse or stop) and speed (via PWM) of a DC motor using buttons. The user can manually change the direction of the fan using these buttons and direct the system according to his/her own wishes. One of the main components of the system is a fan designed to manage indoor temperature and ventilation.

The direction of rotation of the fan (clockwise or counterclockwise) is critical because this direction directly affects the airflow pattern and the overall efficiency of the system. In this project, the fan automatically adjusts its operation depending on the ambient temperature. Specifically, when the ambient temperature exceeds 20°C, the fan automatically starts rotating to the left (counterclockwise). When the temperature drops below 10°C, the fan rotates to the right (clockwise) to provide warmer air circulation.

In addition, the system uses LEDs and a buzzer to notify the user when the room temperature is outside the appropriate range. Thanks to these visual and audible notifications, the user is instantly informed of the ambient conditions.

Correct fan rotation is necessary to provide efficient temperature control, optimize energy consumption, and correctly simulate the real world of the system. Otherwise, the desired airflow cannot be achieved, heat dissipation is impaired, and system performance is significantly reduced. Therefore, the implementation of the correct fan direction is vital for the technical accuracy of the project and the effective management of indoor conditions..

1.2 Features

- **Temperature Measurement and Display:** It uses an LM35 analog temperature sensor and an LCD1602 display to show the ambient temperature in the format of `"Temperature: XX°C"`. The display is automatically updated when a temperature change of ≥ 0.5°C is detected.
- **Dual Mode Fan Operation:** The system is designed to operate in two different modes:
  - **Auto Mode:** When the ambient temperature exceeds 20°C, the fan automatically starts rotating to the left. The fan speed is controlled via PWM. If the temperature drops below 20°C, the fan

automatically stops. If the temperature drops below 10°C, the fan starts rotating to the right to restore optimal conditions.

  - **Manual Mode:** The user can manually control the fan using five push buttons. These buttons allow the user to select the fan direction (forward, reverse, or stop) and adjust the fan speed in ±10 steps by changing the PWM duty cycle.

- **Motor Control:**
  - **Direction Control:** Physical buttons are used to adjust the fan direction (forward, reverse, stop).
  - **Speed Control:** Fan speed is controlled using PWM, which can be adjusted from 0–100% with dedicated increase/decrease buttons.

- **LED and Buzzer Alerts:** Visual and audible alerts are provided to inform the user of the current temperature status:
  - When the temperature exceeds 20°C or drops below 10°C: The fan is activated, the green LED turns off, the red LED lights up, and the buzzer sounds.
  - When the temperature returns to the 10–20°C range: The fan stops, the red LED and buzzer turn off, and the green LED is activated instead.

  This keeps the user constantly informed of the system status through both visual and audio-based indicators.

- **Serial Communication (UART):** The system transmits the current PWM duty cycle via UART in the form of `"PWM:0˚"` whenever the speed changes. This allows external devices or a computer to monitor or log fan performance in real time.

- **EXTI Interrupts:** All button inputs are processed using GPIO external interrupts to ensure responsive and non-blocking control throughout system operation.

2 Hardware Components and Configuration

In this project, a temperature-sensitive fan control system was designed that operates in both automatic and manual modes. The system is built around the STM32F103R6 microcontroller and integrates various sensors, actuators, and user interface components.

Main Hardware Components

- **Microcontroller (STM32F103R6):** Serves as the core unit of the system. It handles temperature acquisition, PWM generation, UART communication, LCD display control, and button interrupts.

- **Temperature Sensor (LM35):** Measures the ambient temperature and provides an analog voltage as output. The STM32's ADC (Analog-to-Digital Converter) reads this voltage and converts it to a temperature value using the formulas below:

$$Voltage = \frac{ADCValue \times 5.0}{4095} \tag{1}$$

$$Temperature(C) = \frac{Voltage \times 100}{2} \tag{2}$$

  This provides a resolution of 10mV per °C, as per LM35 specifications.

- **Motor Driver (L298N):** A dual H-bridge motor driver that allows the DC fan motor to rotate in both forward and reverse directions. PWM input from the microcontroller adjusts the speed, and digital pins control the direction.

- **DC Fan:** The fan operates based on the ambient temperature:
  - Below 10°C: Fan rotates to the right (forward).
  - Between 10°C and 20°C: Fan stops.
  - Above 20°C: Fan rotates to the left (reverse).

- **LCD Display (16x2):** Displays real-time temperature readings and fan status. The display is updated only when there is a significant temperature change (0.5°C), which helps reduce unnecessary refresh operations.

- **Buzzer and LEDs:** The red LED and buzzer activate when the temperature goes beyond the safe threshold (either too low or too high). The green LED indicates a safe temperature range.

- **Push Buttons:** Five push buttons are used to manually control the system:
  - `FWD`, `REV`, `STOP` — control fan direction
  - `PWM+`, `PWM` — increase/decrease fan speed

  These buttons are handled via external interrupt (EXTI) lines for instant response.

- **UART and Virtual Terminal:** The PWM value is transmitted over UART and displayed on a virtual terminal. This feature is useful for debugging and monitoring the system remotely.

Proteus Simulation

Before physical implementation, the complete circuit was designed and tested in **Proteus 8.17** simulation environment. This step ensured that all pin mappings, ADC-PWM behavior, LCD display updates, and UART communication worked correctly.

Initially, an older version (Proteus 8.13) failed to simulate the LM35 sensor properly. After upgrading to version 8.17, the simulation was successful. UART communication with the virtual terminal also functioned correctly after further configuration adjustments.

2.1 Working Principle

When the system is started, all peripherals (GPIO, Timer, ADC, LCD) are initialized and necessary configurations are made.

The fan is stopped at the beginning; the PWM duty cycle is set to 0% by default.

Fan control is carried out in two different modes:

- **Automatic Mode:**
  - When the ambient temperature exceeds 20°C, the fan starts to rotate to the left. The fan speed is adjusted using PWM depending on the temperature.
  - If the temperature drops below 10°C, the fan rotates to the right.
  - The fan stops when the temperature is between 10°C and 20°C.
  - A red LED and a buzzer provide a warning when the temperature is high; when the temperature is low, the green LED turns on.
- **Manual Mode:**
  - The user can control the fan's direction (forward, reverse, stop) and speed (PWM duty cycle) using five push buttons.
  - Each button operates via external interrupt (EXTI).

Whenever the PWM value is changed, the current value is transmitted to the external environment via UART in the format "PWM:XX".

This structure allows the system to automatically respond to temperature changes while also supporting manual control for user intervention.

3 System Design

3.1 System Diagrams

The main logic of the ventilation system is illustrated in the following diagrams. Figure 1 shows the digital control flow of the system and the interactions with the button/sensor. Figure 2 shows the rotation direction of the fan according to summer and winter modes and the effect of these directions on the ambient air.

Figure 1 is a basic control flow diagram showing how the system works from start to finish. At the beginning of the system, the hardware is initialized and then a continuous loop is entered. In this loop:

- The temperature value is calculated by reading analog data from the LM35 temperature sensor.

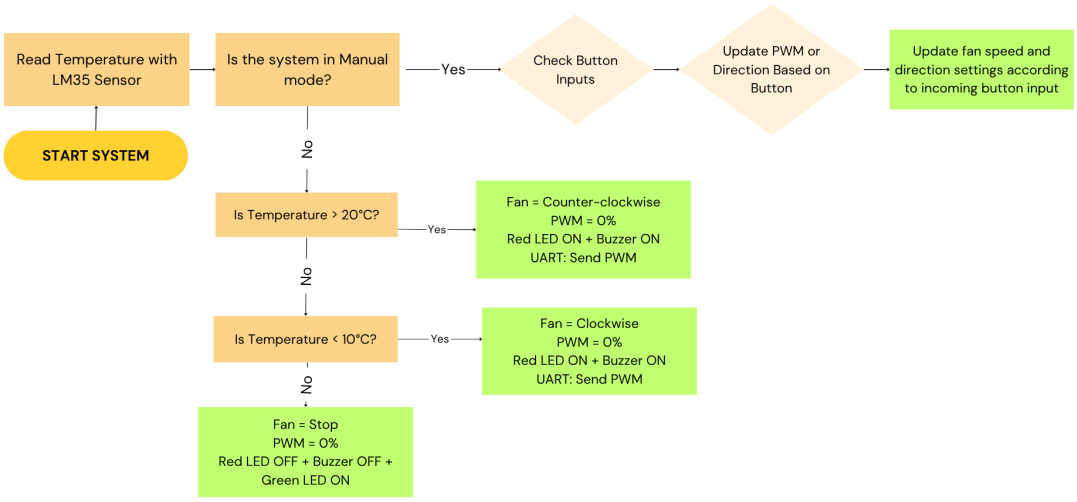

Figure 1: Ventilation System Digital Connections and Control Logic.

Figure 2: Fan Direction Control Mechanism.

- This temperature is compared with the previously recorded value. If the difference exceeds a certain threshold (± 0.5°C), the LCD display is updated.
- The system then checks whether it is in manual or automatic mode:
  - **Manual mode:** Fan direction and PWM value are set by the user based on button inputs.
  - **Automatic mode:** Fan direction, LEDs, and buzzer are controlled by the system depending on the measured temperature.

The automatic control logic is as follows:

- Temperature > 20°C: Fan runs in reverse (summer mode), red LED is ON, and buzzer is activated.
- Temperature < 10°C: Fan runs in forward (winter mode), red LED is ON, and buzzer is activated.
- 10°C $\leq$ Temperature ≤ 20°C: Fan stops, green LED lights up, and buzzer remains off.

This logic enables the ventilation system to work dynamically by responding quickly to changes in ambient temperature.

Figure 2 visually explains how the fan works in different temperature scenarios and the effect of its direction on the ambient air.

Summer Mode (Reverse Rotation / Counterclockwise):

- The fan cools the environment by blowing cool air downwards.
- This mode is automatically activated when the temperature exceeds 20°C.
- System arrangement: PC8 = HIGH, PC9 = LOW → Fan rotates to the left.

Winter Mode (Clockwise):

- The fan circulates the hot air accumulated at the top downwards, ensuring more uniform heating of the environment.
- This mode is manually activated by the user.
- System arrangement: PC8 = LOW, PC9 = HIGH → Fan rotates to the right.

These two rotation modes represent an important design detail that affects not only the fan direction but also the thermal comfort of the environment.

4 Implementation

The hardware for this project is centered around the STM32F103R6 microcontroller. The initial configuration of the peripherals and pin assignments was created using the STM32CubeMX software.

Figure 3 provides a visual representation of the pin configuration for the STM32 microcontroller specified in STM32CubeMX. This setup is critical for defining how the microcontroller interacts with external hardware.

The specific functions of the buttons and a detailed list of their pin connections are explained in more detail in Table 1 and Table 2, respectively.

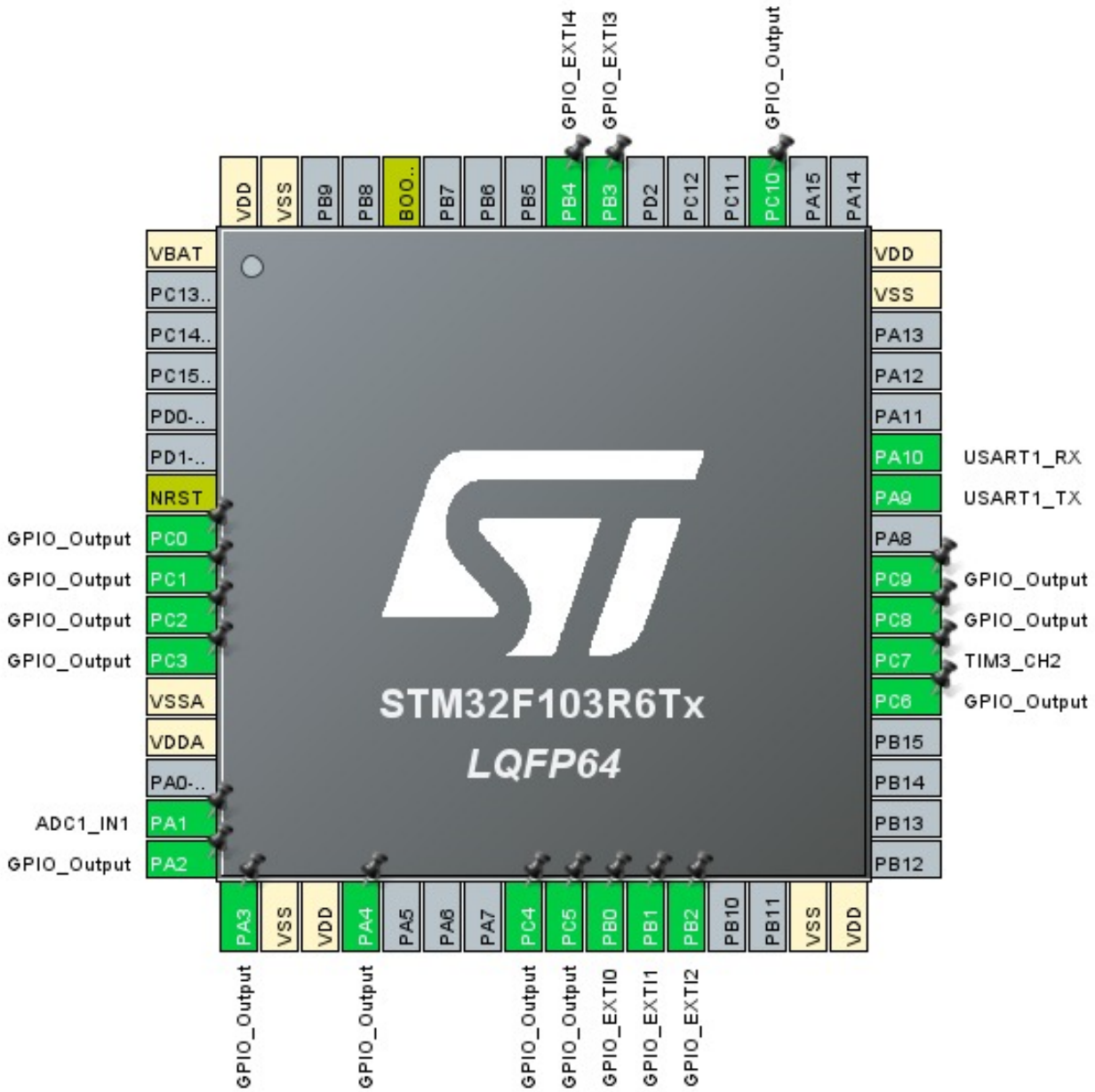

Figure 3: STM32CubeMX Pinout Configuration

Table 1: Button Functions

Button (GPIO)	Function
PA0	Run fan forward
PA1	Run fan reverse
PA2	Stop fan
PA3	PWM duty cycle +10
PA4	PWM duty cycle -10

STM32 Pin	Connected Component / Pin	Description
PA0	Buzzer (Sounder)	Buzzer controlled via PWM output (TIM2_CH1)
PA1	LM35 Temperature Sensor	Analog input for temperature measurement (ADC1_IN1)
PA2	LCD RS (Register Select)	Selects command (0) or data (1) register
PA3	LCD R/W	Read/Write control (kept LOW for write mode)
PA4	LCD E (Enable)	Pulse signal to latch data into LCD
PA9	UART TX (Virtual Terminal)	Sends PWM value over UART to terminal
PB0	Button 1	Run fan forward (Manual mode)
PB1	Button 2	Run fan reverse (Manual mode)
PB3	Button 3	Stop fan (Manual mode)
PB4	Button 4	Increase PWM duty cycle by 10%
PB10	Green LED	Indicates safe temperature (10–20°C)
PB11	Red LED	Indicates danger (Temp <10°C or >20°C)
PC0-PC6 && PC10	LCD D0-D7	LCD Data pin (bit 0-7)
PC7	L298N ENA	PWM input for fan motor speed control (TIM3_CH2)
PC8	L298N IN1	Motor direction control – forward
PC9	L298N IN2	Motor direction control – reverse

Figure 4: Supplementary Configuration Tables.

4.1 Proteus Design and Configuration

After the necessary peripheral unit configurations were made via CubeMX and the software codes were completed in the Keil MDK environment, the circuit design was carried out in the Proteus environment. First, the LM35 temperature sensor was added to the circuit for temperature measurement and connected to the STM32 microcontroller via the analog input pin (PA1 – ADC1_IN1).

In order to display the temperature values to the user, the LCD1602 display was used; the data and control pins were connected to the appropriate GPIO pins (PA2, PA3, PA4, PC0–PC6, PC10) of the STM32, allowing the LCD to operate in 8-bit mode.

A DC motor and L298N motor driver integrated circuit were used for fan control. While the fan speed was controlled from the PC7 (TIM3_CH2) pin with the PWM signal, the direction control was provided via the PC8 and PC9 pins.

A Virtual Terminal (UART) connection was established to transfer information about the system's operation to the user. The Virtual Terminal component was added in Proteus via the PA9 (USART1_TX) pin, and information such as the PWM value is displayed in this terminal using UART communication.

In addition, in order to visually and audibly express the system's reactions to temperature:

- **Red LED (PB11):** Lights up in dangerous temperature ranges (below 10°C or above 20°C).
- **Green LED (PB10):** Lights up in the safe temperature range (10–20°C).

- **Buzzer (PA0):** Provides an audible warning in dangerous situations by being controlled with PWM.

These components have been added to the Proteus circuit to provide instant feedback to the user about the current status of the system. While the LEDs are directly connected to the GPIO output, the buzzer is connected to the PA0 pin to generate a tone frequency with the PWM output.

There are five buttons in the system for manually changing the fan direction and speed (Forward rotation, Reverse rotation, Stop, PWM increase, PWM decrease). These buttons are connected to the pins PB0–PB4 of the STM32 microcontroller and each is configured as an external interrupt (EXTI). In this way, the fan is automatically started when the system exceeds the 20°C temperature threshold, while the user can manually change the direction and speed of the fan in the desired temperature range.

This structure allows the system to provide both automatic and manual control capabilities at the same time, offering a more flexible and user-oriented ventilation experience.

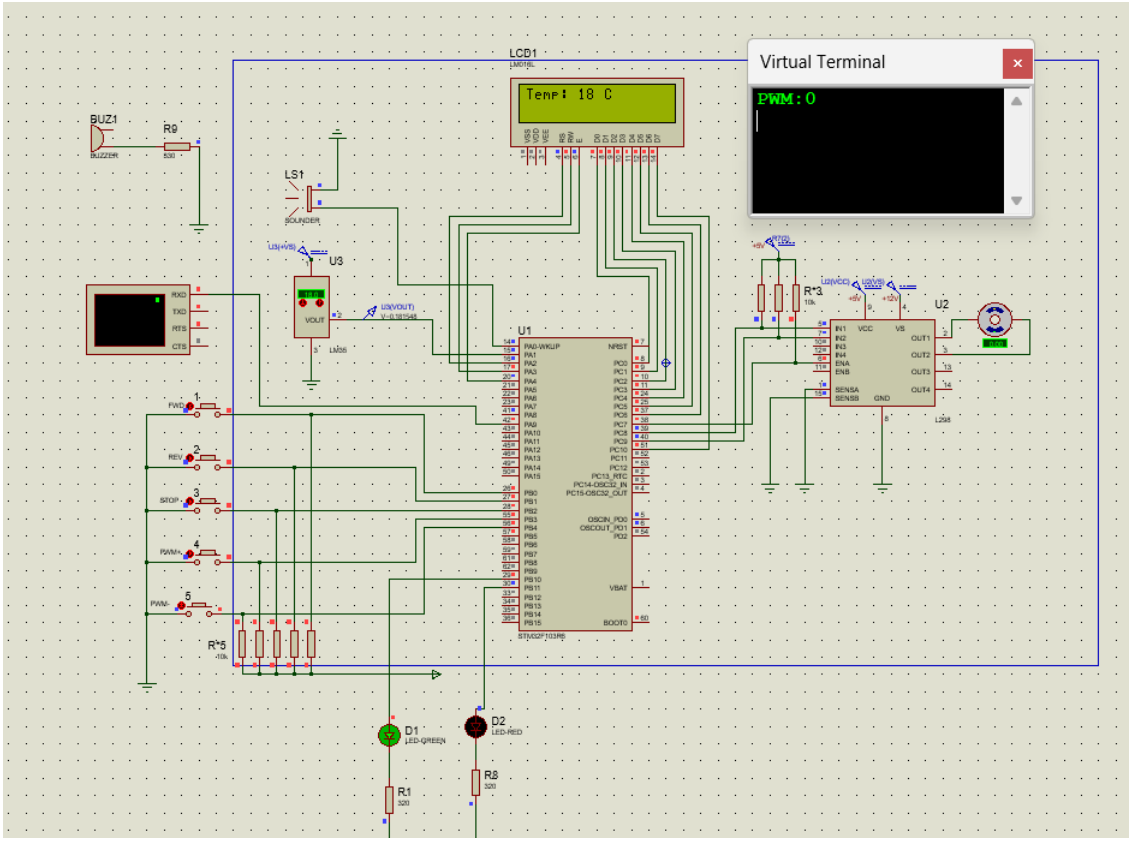

Figure 5: System operating in safe temperature range. The fan remains idle, and the green LED is on.

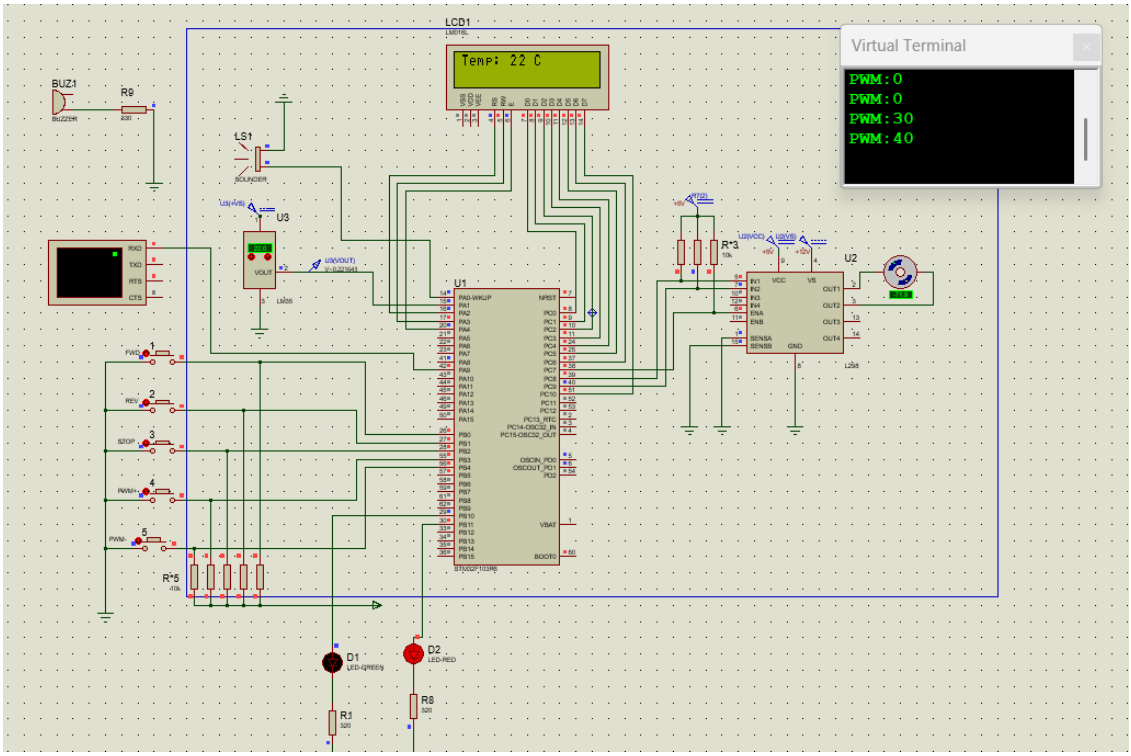

Figure 6: System automatically activated above the safe temperature threshold. At 22°C, the fan rotates left, red LED and buzzer are triggered.

Figure 5 shows the system operating within the safe temperature range (10–20°C). In this state, the fan remains off and the green LED is active. Figure 6 illustrates the behavior of the system when the temperature exceeds 20°C. At 22°C, the system automatically activates the fan in the reverse direction (cooling), with the red LED and buzzer also engaged.

5 Summary Table for Final Report

Table 1: Estimated Price List for Electronic Components (May 2025, Turkey Market)

Component	**Estimated Price (TRY)**
STM32F103R6 Microcontroller (Development board, e.g., Blue Pill)	400 – 600
LM35 Temperature Sensor (Standard type)	20 – 70
LCD 16x2 Display (Character LCD, HD44780 compatible)	70 – 200
L298N Motor Driver (Module board with heatsink)	80 – 250
DC Motor (Hobby Type, small)	30 – 150
Push Buttons (5 pcs, tactile switch)	5 – 25
Buzzer (PWM-controlled sounder module)	10 – 40
LEDs (Red + Green, 5mm)	2 – 10
Resistors (Assorted resistor pack, e.g., 100 pcs)	20 – 60
Other Auxiliary Materials	
Breadboard (Medium prototyping board)	50 – 150
Jumper Wires (M-M, M-F, F-F, assorted set of 40+)	30 – 100
Small Electronic Parts (e.g., potentiometer for LCD contrast)	10 – 30
Power Supply (Adapter or battery, depending on system needs)	100 – 300

Module/Feature	We used these types:
GPIO	⭐ Digital Output ⭐ Digital Input
Communication	⭐ UART
Watchdog timer	Not used
Interactivity (Leds, buttons, switches, touch etc.)	⭐⭐ 5 push buttons → For manual fan control and PWM adjustments (PB0–PB4) ⭐⭐ 2 LEDs → Red (PB11) for danger, Green (PB10) for safe temp ⭐⭐ LCD 16x2 display → PA2, PA3, PA4, PC0–PC6, PC10 (8-bit mode) ⭐⭐ Buzzer → PA0, used with PWM to emit sound at critical temperatures
Using sensors	⭐ Single sensor → LM35 Temperature sensor (analog)
Actuators	DC motor + L298N driver module
Timers	⭐⭐ TIM3 → PWM generation for fan (PC7) ⭐⭐ TIM2 → PWM generation for buzzer (PA0) ⭐⭐ SysTick → Delay & timing calculations
Usage of polling	Not used
Usage of Interrupts	⭐⭐ PB0–PB4 → Configured as external interrupts (EXTI)
Error handling	⭐ In 1 place → Error_Handler () used for ADC and Timer initialization errors.
Analog-digital Converter	⭐⭐ADC1: used to measure voltage from LM35 sensor on PA1. Converted value used for temperature-based fan and alert logic
	Not used
Advanced Things that no code is provided during the course such as DAC, CAN etc.	Not used
Power saving	Not used
DMA	Not used
Ethernet-internet-Wi-Fi	Not used
Writing own driver library for a peripheral	⭐⭐⭐⭐⭐A complete LCD1602 driver library (lcd1602.h) was written from scratch using HAL GPIO functions. It includes pin control macros, data direction handling, command/data writing, busy flag checking, and LCD initialization.
Bluetooth	Not used
PCB	Not used
Usage of advanced tools e.g., Matlab, CubeAI etc. (Matlab code should run on MCU)	Not used
Real time OS	Not used

Figure 7: Module and Feature Use Summary Table.

6 Challenges

During the development phase of the project, a major technical issue was encountered regarding Proteus version compatibility. Initially, the circuit was designed and simulated using Proteus version 8.13. However, in this version, the LM35 temperature sensor did not work properly and could not produce a valid analog output or interact correctly with the STM32 microcontroller.

Despite thorough checking of the wiring, ADC configuration and voltage levels, the problem persisted. After further investigation, it was determined that the problem was most likely caused by software limitations or bugs present in the older Proteus version.

To resolve this, the simulation environment was upgraded to Proteus 8.17. After the update, the LM35 sensor started working properly and the temperature values were successfully read and displayed on the LCD screen.

However, after this upgrade, a new issue arose: the Virtual Terminal (UART communication) stopped working as expected. Initially, despite the correct configuration, no outputs were visible. After careful debugging, adjustments were made to the USART1 initialization settings, Proteus UART module parameters, and Keil code structure.

Thanks to these improvements, UART communication was successfully reestablished. As a result, PWM values are now transmitted correctly and displayed in the Virtual Terminal alongside real-time sensor readings.

7 Team Contributions

All team members equally contributed to the design, development, and debugging stages of the project. Tasks were not assigned strictly to individuals; instead, a collaborative workflow was adopted throughout the process. Each member actively participated in configuring peripherals via STM32CubeMX, writing and testing embedded C code in Keil, and designing and simulating the circuit in Proteus.

During the development, technical issues such as sensor malfunctions and UART communication problems were addressed collectively through group discussion and step-by-step debugging. This approach ensured that all members gained hands-on experience with every stage of the embedded system design and contributed equally to the project's success.