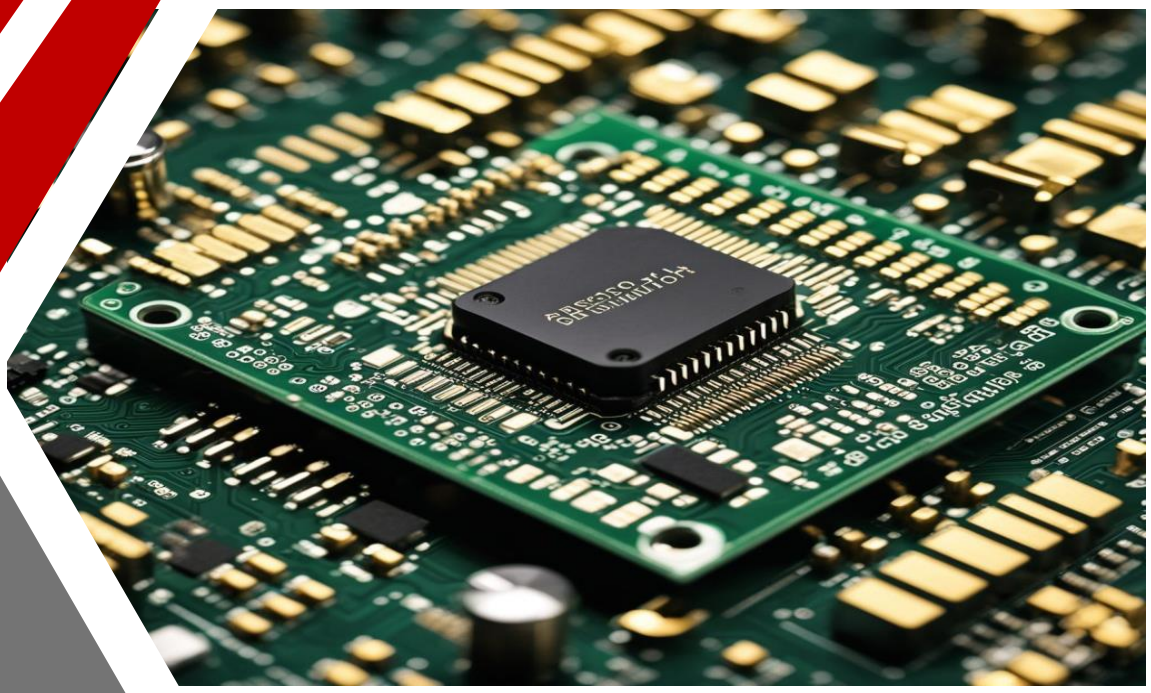
MICROPROCESSORS

METHODOLOGY:

Smart Fan Control System

2025



GROUP 12 MEMBERS

AFFUL ANDREWS QUANSAH – 1863522
YEBOAH NANA SAFO - 1862922
YEBOAH HENRY ASANTE – 1862822
BENJAMIN AKYEN NKETSIAH 1863322
ZIBLIM MARY MMINWUNI – 1863222
ZAKARI ABDUL JELILI - 1863122

Contents

Methodology: Smart Fan Control System

1 Problem-Solving Approach:

The primary objective of this project is to develop an automated fan control system that dynamically adjusts fan speed based on real-time room temperature contributing to smart homes and smart community ecosystems.

1.1 Problem Identification

The challenge lies in accurately measuring temperature, translating it into actionable fan speed commands while maintaining a comfortable environment, and displaying the system's status in a user-friendly manner.

The approach taken to solve this problem involved:

- **Temperature Sensing**: Utilizing a *TMP36* analogue temperature sensor to continuously monitor the ambient room temperature. This sensor was selected for its simplicity, linear output, and ease of integration with the Arduino microcontroller.
- **Microcontroller (Data) Processing**: Employing an *Arduino Uno R3* as the central processing unit to read analogue temperature data from the TMP36, perform calculations, and generate corresponding Pulse Width Modulation (PWM) signals to control the fan's speed.
- **Fan Speed Control**: Implementing a *TIP120 NPN* transistor to regulate the DC motor's speed through PWM. This transistor allows the Arduino's low-current output to control the higher current required by the fan motor.
- **User Interface:** Incorporating a *16x2 LCD* display to provide real-time feedback to the user, displaying the measured temperature and the current fan speed status.
- **Modular Programming**: Structuring the Arduino code into distinct functions and logical blocks to enhance readability, maintainability, and scalability. Basically, we organized the Arduino’s instruction like building blocks to make it easier understand.
- **Iterative Development**: Employing an iterative development process, starting with basic temperature sensing and fan control, and progressively adding features like LCD display and refined control logic.
- **Simulation and Testing**: Utilizing a computer program called Tinkercad for initial circuit simulation and code testing to validate the system's functionality.

2 Code and Schematics Overview:

2.1 Schematics:

The circuit is designed to interface the TMP36 sensor, DC motor, and Arduino. Here's a detailed explanation of the schematic:

- Arduino Uno R3: Serves as the microcontroller, processing sensor data and controlling the fan and LCD. Basically acts as the brain of the system
- TMP36 Temperature Sensor:
  - ⇒ Connected to the Arduino's analogue pin A0 to read analogue temperature values.
  - ⇒ Powered by the Arduino's 5V and GND pins.
- TIP120 Transistor:
  - ⇒ Base pin connected to Arduino digital pin 9 (PWM) through a 1kΩ current-limiting resistor.
  - ⇒ Collector pin connected to the DC motor.
  - ⇒ Emitter pin connected to GND.
- DC Motor:
  - ⇒ One terminal connected to the TIP120's collector.
  - ⇒ The other terminal connected to 5V.
- 16x2 LCD Display:
  - ⇒ RS, E, D4, D5, D6, and D7 pins connected to Arduino digital pins 12, 11, 5, 4, 3, and 2, respectively.
  - ⇒ VSS connected to GND, VDD connected to 5V, and VO connected to a 10kΩ potentiometer for contrast adjustment.
  - ⇒ 10kΩ Potentiometer:
  - ⇒ Centre pin connected to the LCD's VO pin.
  - ⇒ Outer pins connected to 5V and GND for voltage division.

Workflow

- ➢ The TMP36 temperature sensor continuously measures the ambient room temperature.
- ➢ This measurement is an analogue signal.
- ➢ The Arduino Uno's analogue input pin (A0) reads the analogue signal from the temperature sensor.
- ➢ The Arduino's built-in Analog-to-Digital Converter (ADC) converts this analogue signal into a digital value.

- The Arduino's microcontroller processes the digital value.
- It applies a formula to convert the digital value into a temperature reading in degrees Celsius.
- The Arduino compares the calculated temperature to predefined thresholds.
- Based on these comparisons, it determines the appropriate fan speed (off, slow, medium, or fast).
- The Arduino generates a Pulse Width Modulation (PWM) signal.
- The duty cycle of the PWM signal corresponds to the desired fan speed.
- The PWM signal is sent to the TIP120 transistor.
- The TIP120 acts as a switch, controlling the current flow to the DC motor.
- The motor's speed is adjusted according to the PWM signal.
- The Arduino sends the temperature reading and fan speed information to the 16x2 LCD.
- The LCD displays this information to the user.
- Serial Monitoring (Optional):
- The Arduino can also send the temperature and fan speed information to a computer via the Serial Monitor for debugging and monitoring.

Simplified Diagram

[Temperature Sensor (TMP36)] ⟶[Analog -to-digital conversion(Arduino A0)] → [Temperature calculation(Arduino)] → [Fan speed determination(Arduino)] → [PWM Signal generation(Arduino)] → [Motor Control(TIP120)][DC Motor]

↑

|

|

[LCD Display (16x2)]———————— |———————⟶[Arduino]

2.2 Code:

The Arduino code is structured as follows:

- Library Inclusion

[#include <LiquidCrystal.h>:]

⇒ Includes the LiquidCrystal library for LCD control.

- Pin Definitions:

⇒ Defines constants for the temperature sensor pin (tempPin), motor control pin (motorPin), and LCD pins.

- Object Initialization:

⇒ Creates a LiquidCrystal object to control the LCD.

- setup() Function:

⇒ Initializes serial communication for debugging.

⇒ Configures the motor pin as an output.

⇒ Initializes the LCD with lcd.begin(16, 2).

⇒ Prints the initial "Temp: " text to the LCD.

- loop() Function:

⇒ Reads the analogue value from the TMP36 sensor using analogRead().

⇒ Converts the analogue value to Celsius temperature.

⇒ Prints the temperature to the Serial Monitor.

⇒ Implements if-else if-else logic to determine the fan speed based on temperature ranges.

⇒ Generates PWM signals using analogWrite() to control the motor speed.

⇒ Stores the current fan speed in a variable.

⇒ Displays the temperature and fan speed on the LCD using lcd.print() and lcd.setCursor().

⇒ Introduces a delay() for periodic temperature readings and fan speed adjustments.

This methodology provides a comprehensive framework for the development and understanding of the smart fan control system, facilitating its implementation and potential future enhancements.