Design and Implementation of a PWM-Based Smart Temperature-Regulated Fan System Using Arduino

Problem Statement (Part 2 - Moderate Task)

Traditional fan systems lack responsiveness to environmental conditions, resulting in energy inefficiency and poor comfort control. The goal of this project is to build a temperature-aware smart fan system using an Arduino. It dynamically adjusts fan speed using Pulse Width Modulation (PWM) based on ambient temperature. Additionally, a 16x2 LCD provides real-time feedback, and a buzzer offers audio alerts for specific temperature thresholds...

System Design Overview

The system consists of the following key modules:

- Temperature Sensor (TMP36): Continuously monitors ambient temperature and converts it into an analog voltage.
- Arduino Uno R3: Processes sensor input, calculates temperature, maps it to a fan angle using PWM, and controls other peripherals accordingly.
- Servo Motor: Used to simulate a fan with angle-based speed control.
- 16x2 LCD Display: Displays the live temperature and corresponding fan speed percentage.
- Piezo Buzzer: Alerts the user for extreme temperature conditions - such as freezing or overheating.
- Resistors (1kOhm and 220kOhm): Used for signal conditioning and safety.

Purpose of Each Component

Component	Purpose
TMP36 Sensor	Measures temperature and outputs voltage corresponding to temperature.
Servo Motor	Acts as a stand-in for a fan; PWM angle maps to speed.
LCD (16x2)	Shows real-time temperature and fan speed level to the user.
Piezo Buzzer	Emits tones based on specific temperature ranges (e.g., freeze, hot).
Arduino Uno R3	Controls all logic, reads sensor values, and drives output devices.
Resistors (1kOhm, 220kOhm)	Used for protection and voltage division if needed.

Temperature and Device Response Pairing

Temperature (C)	Fan Speed (Servo Angle)	LCD Message	Buzzer Tone
0	0 deg	Temp: freeze	440 Hz
1-10	0 deg	Temp: cool	1000 Hz
11-50	Mapped 0-90 deg	Temp: XX.X C	OFF
>50	90 deg	Temp: hot	2000 Hz + delta T

Explanation of Temperature Response

- Below 1 deg C: Fan remains OFF, LCD shows "freeze," buzzer alerts with low tone (440 Hz).

- 1 deg C to 10 deg C: LCD shows "cool," buzzer alerts with mid-tone (1000 Hz), fan stays off.

- 11 deg C to 50 deg C: Fan angle is mapped linearly from 0 to 90 deg, no buzzer sound, LCD shows exact temperature and fan speed.

- Above 50 deg C: Fan at max speed (90 deg), LCD shows "hot," buzzer plays high-pitched increasing tone based on how far temperature exceeds 50 deg C.

Arduino Code

```
#include <LiquidCrystal.h>
#include <Servo.h>

LiquidCrystal lcd(8, 9, 3, 4, 5, 6);
Servo myServo;
const int tempin = A0;
const int fanpin = 11;
const float lowtemp = 10.0;
const float hightemp = 50.0;
const float freezelevel = 0.0;
const float maxdeg = 90.0;
const float mindeg = 0.0;
const int buzzer = 10;

void setup() {
  lcd.begin(16, 2);
  lcd.print("Temp: ");
  pinMode(tempin, INPUT);
  pinMode(buzzer, OUTPUT);
  myServo.attach(fanpin);
  Serial.begin(9600);
}
```

```
float toCel(float a) {
  return ((a * 5.0 / 1024.0) - 0.5) * 100.0;
}

float setdeg(float t) {
  if (t < lowtemp) {
    return mindeg;
  } else if (t >= lowtemp && t <= hightemp) {
    return map(t, lowtemp, hightemp, mindeg, maxdeg);
  } else {
    return maxdeg;
  }
}

void print(float t, float deg) {
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  if (t <= freezelevel) {
    tone(buzzer, 440);
    lcd.print("freeze");
  } else if (t > freezelevel && t <= lowtemp) {
    tone(buzzer, 1000);
    lcd.print("cool");
  } else if (t >= lowtemp && t <= hightemp) {
    noTone(buzzer);
    lcd.print(t);
    lcd.print("C");
  } else {
    tone(buzzer, 2000 + 2000 * (t - hightemp) / 75);
    lcd.print("hot");
  }

  lcd.setCursor(0, 1);
  lcd.print("Fan Speed:");
  lcd.print((int)(deg * 100 / maxdeg));
  lcd.print("%");
  delay(1000);
  lcd.clear();
}

void loop() {
  myServo.write(0);
  float input = analogRead(tempin);
  float temp = toCel(input);
```

```
  int deg = setdeg(temp);
  myServo.write(deg);
  print(temp, deg);
}
```

TinkerCAD Simulation

https://www.tinkercad.com/things/jVv1RqMzUKs-part-2/editel