**Temperature Controlled Fan with 8051** **Microcontroller**

**Objective:**

The objective of this project is to make a Fan whose speed is controlled by temperature. The idea behind the project is that we have a temperature sensor which senses the temperature and give the output in analog form which then is fed to ADC to convert it to digital signal (with values in HEX format), the output of ADC is given to the microcontroller. Now according to the temperature we are going to control speed of the dc fan which depends on the technique Pulse width modulation, as the width varies, the delay varies as a result of that the speed of the fan varies accordingly.

**Abstract:**

The Temperature Controlled Fan system presents an intelligent solution for effective thermal management in electronic devices. Utilizing a microcontroller-driven approach, this system continuously monitors ambient temperatures through a dedicated sensor and dynamically adjusts the speed of a fan to maintain optimal operating conditions. By responding in real-time to temperature fluctuations, the system ensures energy efficiency, noise reduction, and enhanced longevity of electronic components. With applications ranging from computer systems to industrial equipment, this technology offers a cost-effective and automated means of safeguarding against overheating, thereby optimizing performance and reliability in diverse electronic environments.

**Hardware / Software Requirements:**

**Software :-** Using Proteus 8 Professional

* Microcontroller AT89C52
* ADC0804
* 16\*2 LCD Display
* Temperature Sensor LM35
* Resistors
* Capacitors
* Diodes
* Crystal Oscillator 12MHz
* Electric Fan
* Voltage Regulator (7805)
* Potentiometer (10K)

**Working Principle:**

* The main principle behind the working is PWM i.e. pulse width modulation.
* It is a technique for controlling the power output given to different electric devices.
* The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast pace. The longer the switch is on compared to the off periods, the higher the power supplied to the load is.
* So if the switch is on for a longer time then more power will be delivered to the fan and it will rotate faster.
* We can program the micro controller in such a way that when the temperature is higher, the switch remains on for a longer duration and when it is lower, switch remains on for a shorter duration.

**Flowchart:**

A diagram of a computer system

Description automatically generated

**Approach / Methodology / Programs:**

The approach and methodology for implementing a Temperature Controlled Fan system involve several key steps:

* **Sensor Integration:** The first step is to integrate a temperature sensor into the electronic device or environment requiring thermal management. Common sensors include thermocouples, thermistors, or infrared sensors. The sensor should be strategically placed to provide an accurate representation of the ambient temperature.
* **Microcontroller-Based System:** Employ a microcontroller, such as Arduino or Raspberry Pi, as the central processing unit. The microcontroller reads data from the temperature sensor and processes it to make decisions regarding fan speed control. The programming logic involves defining temperature thresholds and corresponding fan speed levels.
* **Fan Control Mechanism:** Connect the fan to the microcontroller through suitable interfaces, such as pulse-width modulation (PWM) pins. The fan control mechanism should be capable of adjusting the fan speed dynamically based on the temperature readings. This can be achieved through algorithms that map temperature ranges to specific PWM signals, regulating the fan's rotational speed.
* **Feedback Loop:** Establish a feedback loop to continuously monitor the temperature and adjust the fan speed in real-time. This loop ensures that the system remains responsive to temperature changes, preventing overheating and maintaining the desired temperature range.
* **Testing and Calibration:** Conduct thorough testing to validate the system's functionality and responsiveness. Calibration may be necessary to fine-tune the relationship between temperature readings and fan speed adjustments, ensuring accuracy and reliability in diverse operating conditions.
* **Energy Efficiency Considerations:** Implement energy-saving features within the system to optimize power consumption during periods of lower thermal demand. This can involve algorithms that gradually adjust fan speed based on the rate of temperature change, preventing unnecessary fan activity.
* **Integration with Device or System**: Integrate the Temperature Controlled Fan system seamlessly into the target device or environment. This may involve considerations for power supply, physical space, and compatibility with existing electronic systems.

By following these steps, the Temperature Controlled Fan system can be effectively designed and implemented to enhance thermal management in electronic devices, ensuring optimal performance and longevity of critical components.

**Circuit Diagram:-**

A diagram of a circuit board

Description automatically generated

**CODE for Temperature Controlled Fan system with 8051 Microcontroller:-**

1. ORG 0000H
2. LJMP MAIN
3. ORG 000BH
4. LJMP INTERRUPT
5. ORG 0030H
6. **MAIN:**
7. MOV P0, #0FFH
8. SETB P1.5
9. MOV A,#38H
10. ACALL COMMAND
11. ACALL DELAY
12. MOV A,#0EH
13. ACALL COMMAND
14. ACALL DELAY
15. MOV A,#01H
16. ACALL COMMAND
17. ACALL DELAY
18. MOV A,#080H
19. ACALL COMMAND
20. ACALL DELAY
21. LJMP AGAIN1
22. **COMMAND:**
23. MOV P2,A
24. CLR P3.1
25. CLR P3.0
26. SETB P3.2
27. ACALL DELAY
28. CLR P3.2
29. RET
30. **DELAY:**
31. MOV R3,#0FFH
32. AGAIN : DJNZ R3,AGAIN
33. RET
34. **AGAIN1:**
35. MOV A,#' '
36. ACALL LCDWRITE
37. ACALL DELAY
38. MOV A,#'M'
39. ACALL LCDWRITE
40. ACALL DELAY
41. MOV A,#'I'
42. ACALL LCDWRITE
43. ACALL DELAY
44. MOV A,#'C'
45. ACALL LCDWRITE
46. ACALL DELAY
47. MOV A,#'R'
48. ACALL LCDWRITE
49. ACALL DELAY
50. MOV A,#'O'
51. ACALL LCDWRITE
52. ACALL DELAY
53. MOV A,#'C'
54. ACALL LCDWRITE
55. ACALL DELAY
56. MOV A,#'O'
57. ACALL LCDWRITE
58. ACALL DELAY
59. MOV A,#'N'
60. ACALL LCDWRITE
61. ACALL DELAY
62. MOV A,#'T'
63. ACALL LCDWRITE
64. ACALL DELAY
65. MOV A,#'R'
66. ACALL LCDWRITE
67. ACALL DELAY
68. MOV A,#'O'
69. ACALL LCDWRITE
70. ACALL DELAY
71. MOV A,#'L'
72. ACALL LCDWRITE
73. ACALL DELAY
74. MOV A,#'L'
75. ACALL LCDWRITE
76. ACALL DELAY
77. MOV A,#'E'
78. ACALL LCDWRITE
79. ACALL DELAY
80. MOV A,#'R'
81. ACALL LCDWRITE
82. ACALL DELAY
83. MOV A,#0C1H
84. ACALL COMMAND
85. ACALL DELAY
86. MOV A,#'P'
87. ACALL LCDWRITE
88. ACALL DELAY
89. MOV A,#'R'
90. ACALL LCDWRITE
91. ACALL DELAY
92. MOV A,#'O'
93. ACALL LCDWRITE
94. ACALL DELAY
95. MOV A,#'J'
96. ACALL LCDWRITE
97. ACALL DELAY
98. MOV A,#'E'
99. ACALL LCDWRITE
100. ACALL DELAY
101. MOV A,#'C'
102. ACALL LCDWRITE
103. ACALL DELAY
104. MOV A,#'T'
105. ACALL LCDWRITE
106. ACALL DELAY
107. ACALL DELAY1
108. MOV A,#01H
109. ACALL COMMAND
110. ACALL DELAY
111. MOV A,#' '
112. ACALL LCDWRITE
113. ACALL DELAY
114. MOV A,#'T'
115. ACALL LCDWRITE
116. ACALL DELAY
117. MOV A,#'E'
118. ACALL LCDWRITE
119. ACALL DELAY
120. MOV A,#'M'
121. ACALL LCDWRITE
122. ACALL DELAY
123. MOV A,#'P'
124. ACALL LCDWRITE
125. ACALL DELAY
126. MOV A,#' '
127. ACALL LCDWRITE
128. ACALL DELAY
129. MOV A,#'C'
130. ACALL LCDWRITE
131. ACALL DELAY
132. MOV A,#'O'
133. ACALL LCDWRITE
134. ACALL DELAY
135. MOV A,#'N'
136. ACALL LCDWRITE
137. ACALL DELAY
138. MOV A,#'T'
139. ACALL LCDWRITE
140. ACALL DELAY
141. MOV A,#'R'
142. ACALL LCDWRITE
143. ACALL DELAY
144. MOV A,#'O'
145. ACALL LCDWRITE
146. ACALL DELAY
147. MOV A,#'L'
148. ACALL LCDWRITE
149. ACALL DELAY
150. MOV A,#' '
151. ACALL LCDWRITE
152. ACALL DELAY
153. MOV A,#'F'
154. ACALL LCDWRITE
155. ACALL DELAY
156. MOV A,#'A'
157. ACALL LCDWRITE
158. ACALL DELAY
159. MOV A,#'N'
160. ACALL LCDWRITE
161. ACALL DELAY
162. LJMP AGAIN2
163. **LCDWRITE:**
164. MOV P2,A
165. SETB P3.1
166. CLR P3.0
167. SETB P3.2
168. ACALL DELAY
169. CLR P3.2
170. RET
171. **DELAY1:**
172. MOV R3,#0FFH
173. HERE1: MOV R5,#0FFH
174. HERE2: MOV 75H,#02FH
175. HERE3: DJNZ 75H,HERE3
176. HERE4: DJNZ R5,HERE2
177. DJNZ R3,HERE1
178. RET
179. **AGAIN2:**
180. SETB P3.5
181. SETB P3.3
182. CLR P3.4
183. SETB P3.4
184. HERE5: JB P3.5,HERE5
185. CLR P3.3
186. MOV A,#0C1H
187. ACALL COMMAND
188. ACALL DELAY
189. MOV TMOD,#02H
190. MOV IE,#82H
191. MOV R1,P0
192. MOV A,R1
193. MOV R4,A
194. ACALL COMPARE
195. MOV A,R4
196. LCALL CONVERSION
197. LCALL LCDWRITETMP
198. ACALL DELAY1
199. LJMP MAIN
200. **COMPARE:**
201. CLR C
202. CJNE R1,#35,GAIN
203. GAIN: JNC GAIN1
204. CLR C
205. CJNE R1,#25,GAIN2
206. GAIN2: JNC GAIN3
207. CLR TR0
208. LJMP GAIN4
209. GAIN1: ACALL GREATER
210. LJMP GAIN4
211. GAIN3: ACALL LOWER
212. GAIN4: CLR C
213. RET
214. **GREATER:**
215. CLR TR0
216. MOV R2,#0AAH
217. MOV TH0,#0FFH
218. SETB TR0
219. RET
220. **LOWER:**
221. CLR TR0
222. MOV R2,#0AAH
223. MOV TH0,#1FH
224. SETB TR0
225. RET

1. **CONVERSION:**
2. MOV B,#10
3. DIV AB
4. MOV R7,B
5. MOV B,#10
6. DIV AB
7. MOV R6,B
8. MOV A,R6
9. ADD A,#30H
10. MOV R6,A
11. MOV A,R7
12. ADD A,#30H
13. MOV R7,A
14. RET
15. **LCDWRITETMP:**
16. MOV A,R6
17. ACALL LCDWRITE
18. ACALL DELAY
19. MOV A,R7
20. ACALL LCDWRITE
21. ACALL DELAY
22. MOV A,#'C'
23. ACALL LCDWRITE
24. ACALL DELAY
25. RET
26. **INTERRUPT:**
27. CPL P1.5
28. CLR TR0
29. MOV 76H,R2
30. HERER: DJNZ 76H,HERER
31. SETB TR0
32. CPL P1.5
33. RETI
34. END

**SIMULATION OUTPUT OBTAINED:-**

A diagram of a circuit board

Description automatically generated

The simulation output for a Temperature Controlled Fan system typically involves data and visualizations that demonstrate the system's performance in response to varying temperatures.

**Temperature vs. Time Graphs:** Display the simulated ambient temperature over time. This graph should showcase fluctuations in temperature and how the system responds by adjusting the fan speed accordingly. This visual representation is crucial for evaluating the system's ability to maintain the desired temperature range.

**Fan Speed vs. Temperature Graphs**: Illustrate the correlation between the temperature readings and the corresponding fan speed adjustments. This graph provides insights into the system's control logic and its effectiveness in modulating the fan speed based on temperature variations.

**Energy Consumption Analysis:** Simulate and present data on energy consumption, showcasing how the system optimizes power usage during periods of lower thermal demand. This analysis is essential for assessing the system's energy efficiency and potential cost savings.

**Stability and Responsiveness Metrics:** Evaluate the system's stability by examining how quickly and accurately it responds to changes in temperature. Metrics such as settling time and overshoot can be valuable in assessing the overall performance and robustness of the Temperature Controlled Fan system.

**Project Photos:**



**Conclusion:**

In conclusion, the Temperature Controlled Fan system stands as a pivotal advancement in thermal management for electronic devices, offering a sophisticated and automated solution to address the challenges of temperature regulation. By seamlessly adapting fan speed in response to changing ambient conditions, the system not only optimizes energy consumption and reduces noise levels but also plays a crucial role in preserving the integrity and longevity of sensitive electronic components. Its versatility and applicability across various domains underscore its significance as a cost-effective and intelligent tool in ensuring sustained performance and reliability in electronic systems.

**References:**

[**https://www.elprocus.com/temperature-controlled-dc-fan-using-microcontroller/#:~:text=The%20proposed%20system%20temperature%20controlled,%2C%20power%20supply%2C%20operational%20amplifier.**](https://www.elprocus.com/temperature-controlled-dc-fan-using-microcontroller/#:~:text=The%20proposed%20system%20temperature%20controlled,%2C%20power%20supply%2C%20operational%20amplifier.)