Temperature Measurement Using a Thermistor and Raspberry Pi

Introduction:

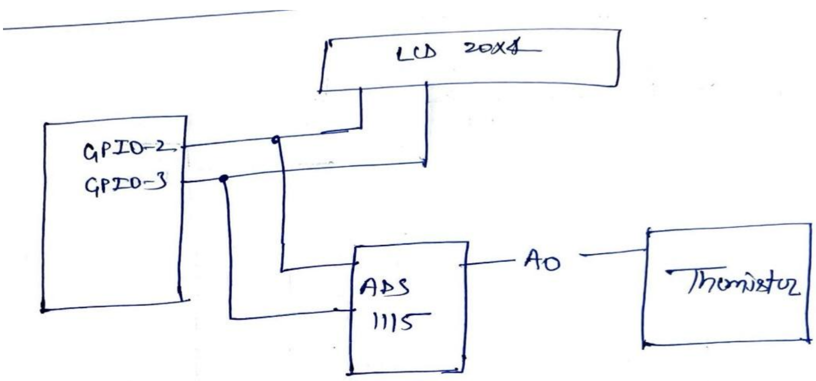
Temperature measurement is crucial in various applications such as weather monitoring, industrial automation, and home automation. A thermistor is a temperature-sensitive resistor that changes resistance with temperature variations. By interfacing a thermistor with a Raspberry Pi, we can measure temperature digitally and use the data for further processing or analysis. This project demonstrates how to use a thermistor with a Raspberry Pi to monitor temperature accurately.

Components Required:

- Raspberry Pi (any model with GPIO support)
- Thermistor (NTC or PTC)
- Resistor (10kΩ, for voltage divider)
- Analog-to-Digital Converter (ADC) module (e.g., MCP3008)
- Breadboard and jumper wires
- Power supply for Raspberry Pi

Circuit Connection:

1. Connect one leg of the thermistor to a 3.3V or 5V power supply.
2. Connect the other leg of the thermistor to one terminal of the 10kΩ resistor.
3. Connect the other terminal of the resistor to the ground (GND).
4. The junction between the thermistor and resistor is connected to an input channel of the ADC.
5. The ADC module is connected to the Raspberry Pi using SPI communication.
6. Once wired correctly, the Raspberry Pi reads voltage values from the ADC and converts them into temperature values using a calibration equation.



(Thermistor)

Applications:

- Environmental monitoring systems
- Industrial and laboratory temperature control
- Smart home automation for HVAC control
- Health and safety applications
- Weather forecasting systems

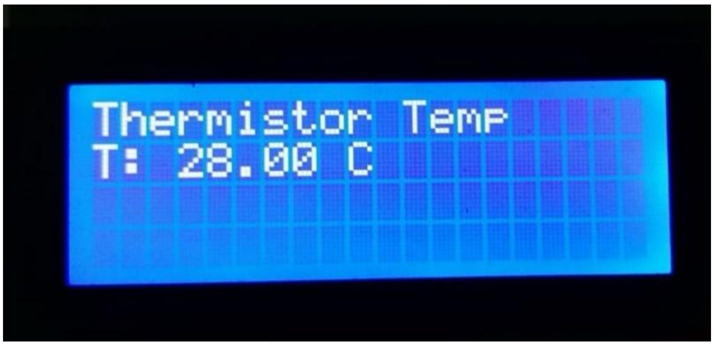
Learnings:

- Understanding the working principle of a thermistor
- Learning how to interface an ADC with Raspberry Pi
- Gaining knowledge about voltage dividers and resistance-temperature conversion
- Hands-on experience with sensor-based data acquisition
- Basics of SPI communication in Raspberry Pi

Conclusion:

This project demonstrates a simple yet effective way to measure temperature using a thermistor and a Raspberry Pi. By using an ADC module, we can convert analog temperature readings into digital values for analysis. This method provides accurate temperature monitoring and can be extended to real-world applications like automated climate control and IoT-based temperature logging systems. Understanding this process opens the door to more complex sensor-based projects with Raspberry Pi.

Outcome:



Program:

```
import time
import board
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
from RPLCD.i2c import CharLCD
import math

# Initialize I2C Bus
i2c_bus = busio.I2C(board.SCL, board.SDA)

# Initialize ADS1115
ads = ADS.ADS1115(i2c_bus)
channel = AnalogIn(ads, ADS.P0)  # Using A0 for thermistor

# Initialize LCD
lcd = CharLCD('PCF8574', 0x27)

# Thermistor Parameters
R_REF = 10000  # Reference resistor (10kΩ)
```

```
BETA = 3950  # Beta coefficient (Check datasheet)
T0 = 298.15  # Room temperature in Kelvin (25°C = 298.15K)

def read_temperature():
   """Calculate temperature from thermistor resistance."""
   Vout = channel.voltage  # Read voltage
   R_thermistor = R_REF * (3.3 / Vout - 1)  # Calculate resistance
   temperature_kelvin = 1 / ((1 / T0) + (math.log(R_thermistor / R_REF) / BETA))
   temperature_celsius = temperature_kelvin - 273.15  # Convert to Celsius
   return temperature_celsius

def update_display(temperature):
   """Update the LCD with temperature."""
   lcd.clear()
   lcd.cursor_pos = (0, 0)
   lcd.write_string("Thermistor Temp")
   lcd.cursor_pos = (1, 0)
   lcd.write_string(f"T: {temperature:.2f} C")

try:
   while True:
      temp = read_temperature()  # Read temperature
      update_display(temp)  # Update LCD
      time.sleep(1)
except KeyboardInterrupt:
   lcd.clear()
   print("Program Stopped")
```