**Data Analysis of Temperature and Humidity Using Arduino and Pandas**

**Abstract**

This report documents the development of a project aimed at analyzing temperature and humidity data using an Arduino microcontroller and Python’s Pandas library. The project integrates hardware and software components to monitor environmental conditions, collect data, and perform detailed analysis to derive meaningful insights. The report elaborates on the motivation, methodology, results, and potential applications of this system.

**Table of Contents:**

1. Introduction
2. Objectives
3. Materials and Components
4. Methodology
5. Hardware Design
6. Software Implementation
7. Data Acquisition
8. Data Analysis Using Pandas
9. Results and Discussion
10. Challenges and Solutions
11. Applications
12. Conclusion
13. References

**1. Introduction**

Climate monitoring is essential for various fields, including agriculture, meteorology, and smart home systems. Accurate and real-time measurement of temperature and humidity can help optimize resources and ensure safety. This project explores how low-cost sensors and open-source software can be combined to create a comprehensive data analysis system.

**2. Objectives**

The primary objectives of this project are:

1. To design a system for monitoring temperature and humidity using Arduino.
2. To collect and store the data for analysis.
3. To analyze the collected data using Python’s Pandas library.
4. To visualize trends and patterns in the environmental data.
5. To explore potential applications of the system.

**3. Materials and Components**

**Hardware:**

1. Arduino Uno
2. DHT11/DHT22 temperature and humidity sensor
3. USB cable
4. Breadboard and jumper wires
5. Power supply

**Software:**

1. Arduino IDE
2. Python (with Pandas, Matplotlib, and other libraries)
3. Jupyter Notebook
4. Serial communication library (e.g., PySerial)

**4. Methodology**

The project is divided into four main phases:

1. **Hardware Setup**: Connecting the sensor to the Arduino and ensuring accurate readings.
2. **Data Collection**: Programming the Arduino to read data and send it to a computer via serial communication.
3. **Data Analysis**: Storing the data in CSV format and analyzing it using Pandas.
4. **Visualization**: Generating graphs and charts to represent trends.

**5. Hardware Design**

The hardware design involves connecting a DHT11/DHT22 sensor to the Arduino. The sensor is used for capturing temperature and humidity. The Arduino reads the data and transmits it to a connected computer.

**Circuit Diagram**

The DHT sensor has three pins:

1. VCC: Connect to 5V.
2. GND: Connect to ground.
3. Data: Connect to a digital pin on the Arduino (e.g., D2).

Arduino Uno Pinout:

1. 5V -> VCC of DHT11
2. GND -> GND of DHT11
3. D2 -> DATA of DHT11

**6. Software Implementation**

**6.1 Arduino Code**

The Arduino is programmed using the Arduino IDE. The DHT library is used to read data from the sensor.

**Sample Code:**

#include <DHT.h>

#define DHTPIN 2

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

dht.begin();

}

void loop() {

float temp = dht.readTemperature();

float hum = dht.readHumidity();

if (isnan(temp) || isnan(hum)) {

Serial.println("Error reading from sensor");

return;

}

Serial.print("Temperature: ");

Serial.print(temp);

Serial.print(" °C, Humidity: ");

Serial.print(hum);

Serial.println(" %");

delay(2000);

}

**6.2 Python Code**

Data from the Arduino is read and stored in a CSV file using Python's PySerial library.

**Sample Code:**

import serial

import csv

from datetime import datetime

arduino = serial.Serial('COM3', 9600)

data\_file = open('temp\_humidity\_data.csv', mode='w', newline='')

csv\_writer = csv.writer(data\_file)

csv\_writer.writerow(['Timestamp', 'Temperature (°C)', 'Humidity (%)'])

try:

while True:

line = arduino.readline().decode('utf-8').strip()

if "Temperature" in line:

parts = line.split(",")

temp = float(parts[0].split(":")[1].strip())

hum = float(parts[1].split(":")[1].strip())

timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

csv\_writer.writerow([timestamp, temp, hum])

except KeyboardInterrupt:

data\_file.close()

arduino.close()

**7. Data Acquisition**

Data is collected over several days under different environmental conditions. The sensor outputs temperature and humidity readings every two seconds, which are stored in a CSV file for analysis.

**8. Data Analysis Using Pandas**

The Pandas library is used for data manipulation and analysis. Below are the key steps:

1. **Loading Data**:

import pandas as pd

data = pd.read\_csv('temp\_humidity\_data.csv')

print(data.head())

1. **Descriptive Statistics**:

print(data.describe())

1. **Data Cleaning**:

Handle missing or inconsistent data.

data.dropna(inplace=True)

1. **Visualization**:

Using Matplotlib for trend analysis.

import matplotlib.pyplot as plt

plt.plot(data['Timestamp'], data['Temperature (°C)'], label='Temperature')

plt.plot(data['Timestamp'], data['Humidity (%)'], label='Humidity')

plt.legend()

plt.show()

**9. Challenges and Solutions**

1. **Sensor Calibration**: Ensured accuracy by comparing readings with standard devices.
2. **Data Noise**: Smoothed noisy data using rolling averages in Pandas.
3. **Connection Issues**: Improved stability by ensuring firm connections.

**10. Applications**

1. **Agriculture**: Monitoring conditions in greenhouses.
2. **Weather Stations**: Setting up low-cost monitoring systems.
3. **Smart Homes**: Automating climate control.

**11. Conclusion**

This project demonstrates the integration of hardware and software to analyze environmental data effectively. The use of Arduino for data collection and Pandas for analysis offers a low-cost yet powerful solution for climate monitoring applications.

**12. References**

1. Arduino Documentation: [https://www.arduino.cc](https://www.arduino.cc/)
2. Python Pandas Documentation: [https://pandas.pydata.org](https://pandas.pydata.org/)
3. DHT Sensor Datasheet: [DHT11 Datasheet PDF - Humidity & Temperature Sensor](https://www.datasheetcafe.com/dht11-datasheet-sensor/)
4. Matplotlib Documentation: [https://matplotlib.org](https://matplotlib.org/)